
# NAVAL POSTGRADUATE SCHOOL

**MONTEREY, CALIFORNIA**

# THESIS

**INVESTIGATING THE IMPLEMENTATION OF KNOWLEDGE REPRESENTATION IN THE COMBATXXI SYSTEM**

by

Mongi Bellili

June 2015

Thesis Advisor: Imre Balogh
Co-Advisor: Curtis Blais

**This thesis was performed in cooperation with the MOVES Institute**

THIS PAGE INTENTIONALLY LEFT BLANK

# REPORT DOCUMENTATION PAGE

*Form Approved OMB No. 0704–0188*



| **1. AGENCY USE ONLY** *(Leave blank)* | **2. REPORT DATE** June 2015 | **3. REPORT TYPE AND DATES COVERED** Master's Thesis |
|---|---|---|

**4. TITLE AND SUBTITLE**
INVESTIGATING THE IMPLEMENTATION OF KNOWLEDGE REPRESENTATION IN THE COMBATXXI SYSTEM

**5. FUNDING NUMBERS**
GM10331601, National Institute of General Medical Sciences of the United States National Institutes of Health

**6. AUTHOR(S)** Mongi Bellili

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Naval Postgraduate School
Monterey, CA 93943-5000

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
N/A

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES** The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.

**12a. DISTRIBUTION / AVAILABILITY STATEMENT**
Approved for public release; distribution is unlimited

**12b. DISTRIBUTION CODE**
A

**13. ABSTRACT (maximum 200 words)**

Combat models and simulations aim to find a balance between complexity and simplicity: Both oversimplification and too much detail can lead to erroneous findings. In simulations that require representation of human behavior, modelers rely on prior scripting to find the balance. However, this technique cannot depict dynamic behavior during the simulation run. This inadequate representation of entity behavior can cause misleading or incomplete results. This thesis investigates the implementation of knowledge representation in combat models in order to enhance entity behavior. The new method does not try to include more details in the model than the scripting method, but it tries to enhance the entity decision making to create more realistic outcomes. A knowledge base along with reasoning capabilities was linked to a combat model to mimic the memory and the brain of an entity. To demonstrate the feasibility of this approach, an ontology development tool called Protégé was linked to a combat model called COMBATXXI. Besides achieving dynamic behavior, the new method has other advantages over previous approaches such as better separation of specification and implementation, loosely-coupled components to allow code reuse, use of well-established reasoners for free, and exploitation of partially-sensed information.

| **14. SUBJECT TERMS** Human Behavior, Scripting, Dynamic Behavior, Knowledge Representation, Ontology, Protégé, COMBATXXI | **15. NUMBER OF PAGES** 117 |
|---|---|
| | **16. PRICE CODE** |

| **17. SECURITY CLASSIFICATION OF REPORT** | **18. SECURITY CLASSIFICATION OF THIS PAGE** | **19. SECURITY CLASSIFICATION OF ABSTRACT** | **20. LIMITATION OF ABSTRACT** |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | UU |

NSN 7540–01-280-5500

Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

**Approved for public release; distribution is unlimited**

# INVESTIGATING THE IMPLEMENTATION OF KNOWLEDGE REPRESENTATION IN THE COMBATXXI SYSTEM

Mongi Bellili
Captain, Tunisian Army
B.S., United States Military Academy, 2008

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN COMPUTER SCIENCE**

From the

**NAVAL POSTGRADUATE SCHOOL**
**June 2015**

Author: Mongi Bellili

Approved by: Imre Balogh
Thesis Advisor

Curtis Blais
Co-Advisor

Peter J. Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

# ABSTRACT

Combat models and simulations aim to find a balance between complexity and simplicity. Both oversimplification and too much detail can lead to erroneous findings. In simulations that require representation of human behavior, modelers rely on prior scripting to find the balance. However, this technique cannot depict dynamic behavior during the simulation run. This inadequate representation of entity behavior can cause misleading or incomplete results. This thesis investigates the implementation of knowledge representation in combat models in order to enhance entity behavior. The new method does not try to include more details in the model than the scripting method, but it tries to enhance the entity decision making to create more realistic outcomes. A knowledge base along with reasoning capabilities was linked to a combat model to mimic the memory and the brain of an entity. To demonstrate the feasibility of this approach, an ontology development tool called Protégé was linked to a combat model called COMBATXXI. Besides achieving dynamic behavior, the new method has other advantages over previous approaches, such as better separation of specification and implementation, loosely-coupled components to allow code reuse, use of well-established reasoners for free, and exploitation of partially-sensed information.

THIS PAGE INTENTIONALLY LEFT BLANK

# TABLE OF CONTENTS

# LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| API | Application Programming Interface |
| BSL | Behavior Specification Language |
| COMBATXXI | Combined Arms Analysis Tool for the 21st Century |
| DES | Discrete Event Simulation |
| DL | Description Logics |
| DM | Decision Module |
| DMI | Decision Module Interaction |
| FM | Functionality Module |
| HTN | Hierarchical Task Network |
| HTTP | Hyper Text Transfer Protocol |
| IDE | Integrated Development Environment |
| IED | Improvised Explosive Device |
| IRI | internationalized Resource Identifier |
| JAR | Java Archive |
| J3CIEDM | Joint Consultation Command and Control Information Exchange Data Model |
| KA | Knowledge Acquisition |
| KB | Knowledge Base |
| KR | Knowledge Representation |
| M-COP | Mobility Common Operational Picture |
| NPS | Naval Postgraduate School |
| OKBC | Open Knowledge Base Connectivity |
| OWL | Web Ontology Language |
| PIED | Packaged IED |
| QL | Query Language |
| RDF | Resource Description Framework |
| RPC | Remote Procedure Call |
| SBIED | Suicide-Bomber IED |
| SITS | Scenario Integration Tool Suite |
| SOAP | Simple Object Access Protocol |

| | |
|---|---|
| SPARQL | SPARQL Protocol and RDF Query Language |
| SWRL | Semantic Web Rule Language |
| TRAC | United States Army TRADOC Analysis Center |
| TRADOC | United States Army Training and Doctrine Command |
| UDDI | Universal Description, Discovery and Integration |
| URI | Uniform Resource Identifier |
| VBIED | Vehicle-Borne IED |
| W3C | World Wide Web Consortium |
| WSDL | Web Services Description Language |
| WSMR | White Sands Missile Range |
| XML | Extensible Markup Language |

# ACKNOWLEDGMENTS

I want to thank Dr. Imre Balogh and Mr. Curtis Blais for their assistance and guidance. Their enthusiasm for the subject kept me focused throughout the process. Also, I would like to thank the faculty and the staff at the MOVES Institute and the Computer Science Department for providing a great learning experience and a place to work.

Finally, I would like to thank my wife and daughter for their patience and understanding.

This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

THIS PAGE INTENTIONALLY LEFT BLANK

# I. INTRODUCTION

## A. BACKGROUND

Reasoning, in computer science, is different from automation and iterative computation. It is a complex operation that aims to mimic the human brain, and it is often accompanied by a knowledge representation scheme to mimic the human memory. Systems that use knowledge representation and reasoning are called knowledge-based systems, and they are favored over conventional procedural techniques by their simplicity and separation of knowledge from reasoning (Smith, 1985). It is not sufficient for entities in modeling and simulation systems to rely solely on iterative and selective computation or, more specifically, on fixed logic and scripted behaviors.

Combined Arms Analysis Tool for the 21st Century (COMBATXXI) is an analytical simulation used to support acquisition and other studies. It is a closed form simulation where the different entities in the model do not receive user input during the simulation run. Currently, entities behave and make decisions based on coded script and their attributes that are stored in the data base. This method is inefficient because the script has to be very detailed. Moreover, this method does not provide realistic behavior because entities lack dynamic reasoning as their knowledge of the battle space grows and changes. Therefore, there is a need to implement artificial intelligence in the COMBATXXI system. Specifically, the actions of the entities should depend on what the system believes and not only on what is explicitly represented. That is, entities should use a reasoner to make use of implicit knowledge in decision making.

## B. RESEARCH PROBLEM

The primary research question is determining the feasibility of using a knowledge representation system with inference capability in a closed form combat simulation (such as COMBATXXI) to support dynamic decision making.

The subsidiary questions are:

- How can a knowledge representation scheme be implemented in COMBATXXI?

- How can the knowledge base be modified during simulation execution?
- How can new knowledge inferred from simulation data affect entity decisions and influence entity behaviors.

## C. OBJECTIVES

In order to better represent knowledge in COMBATXXI, we identify the following objectives:

- Describe a suitable knowledge representation scheme for COMBATXXI, and find a way to connect COMBATXXI with that knowledge representation scheme.
- Find example areas of COMBATXXI that can be represented with the new methodology.
- Determine the type of reasoning that needs to be accomplished in COMBATXXI.
- Construct ontology and implement it in a software application.
- Develop a way to pass data between COMBATXXI and the software application.

## D. THESIS ORGANIZATION

Chapter I gives the motivation of the thesis as well as a general idea how to answer the research question.

Chapter II provides an overview of the simulation system (COMBATXXI) and its parts. Second, it provides a brief history of knowledge representation and its applications. Third, it introduces ontology development and Protégé.

Chapter III details the methodology used to design the technical approach for addressing the research questions. We show the reader, at the conceptual level, how COMBATXXIcan be connected to an Ontology.

Chapter IV describes the implementation of the idea given in chapter III in detail. It gives the reader the steps to recreate the study. In addition, we discuss the results and comment on their relationship to the research question.

Chapter V provides overall conclusions of the research and a discussion of possible future work.

THIS PAGE INTENTIONALLY LEFT BLANK

# II. LITERATURE REVIEW

## A. COMBATXXI

### 1. System Overview

#### *a. Architecture*

COMBATXXI is the Combined Arms Analysis Tool for the 21st Century developed by the TRADOC Analysis Center-White Sands Missile Range (TRAC-WSMR) and the Marine Corps Combat Development Command. It is a closed-form discrete-event simulation at the brigade and lower levels with the intent to model ground combat, air mobile forces, amphibious operations, logistics and casualty handling, and urban operations (Balogh et al., 2014).

Figure 1 shows that COMBATXXI can be decomposed into five major components: the preprocessing tool, the databases, run manager, the simulation engine, and the output tools. First, the preprocessing tool is an interface that allows construction of scenarios. It is called SITS (scenario integration tool suite). Second, there are two databases. The COMBATXXI database is a performance and configuration database, and it contains the data used by all of the models in the simulation. Its data comes from authorized sources and subject matter experts. The second database is the operation and planning database which is specific to the scenario being built. It receives its data from those who are building the scenario. Third, the run manager keeps track of input, output, multiprocessing, and the random seed during the production mode. Fourth, the output tools are used for playback display and post-processing analysis. Finally, the simulation engine is the core component of the system that executes the scenario. Thus, it handles the discrete event queue, timing, and simulation logic. In addition, it describes the environment and the support services such as information access, data logging, and entity interaction control. The system also has the capability to reuse critical scenario components through the library tools.

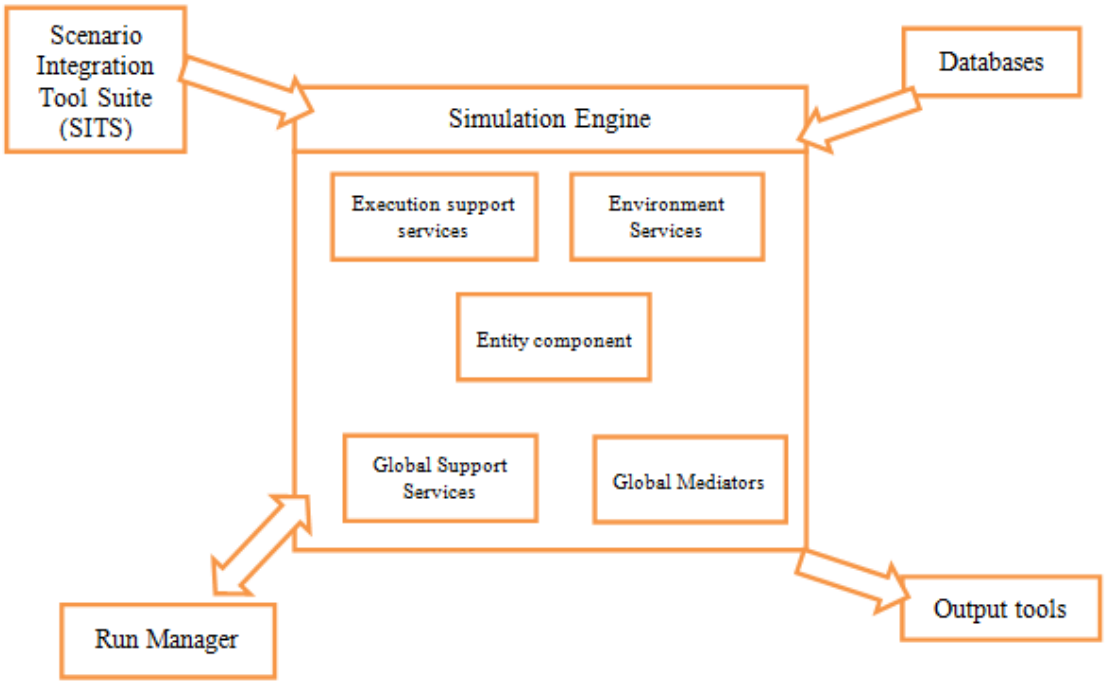


Figure 1. COMBATXXI architecture

### *b. Entities and Behaviors*

Entities are the agents in the model (see Figure 2). Some examples are tanks, aircraft, and soldiers. An entity has several characteristics such as name, kind, domain, profile, state, actions, and behaviors. This necessary data is received from the scenario's XML file and the configuration database. The entities interact with each other and the environment through functionality and decision modules (FMs and DMs, respectively). Only FMs have access to an entity's ground truth. FMsalso model physical interactions and connect the entity to simulation services.

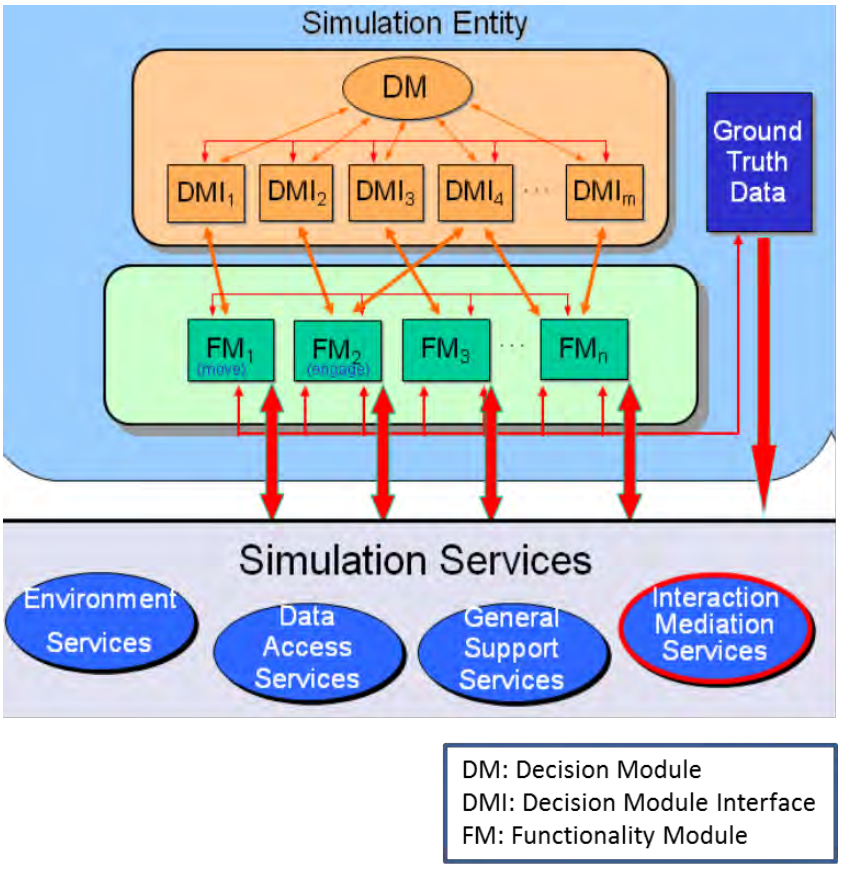


Figure 2. Entity composition and external interactions (from Balogh et al., 2014)

Behaviors are user defined action sequences that control an entity's reaction to some conditions. The behaviors are activated by trigger events during the simulation. Behaviors can be created using three methods: Compound Orders, Behavior Specification Language (BSL), and Python Scripts. Compound Orders are the easiest to use, but they produce static behaviors. They are composed of primitive orders which are the lowest level commands to create behaviors in COMBATXXI. The BSL is the native scripting language in COMBATXXI. On the other hand, Python is an external scripting language. It creates more complex and dynamic behaviors by giving the entities some memory, but it is the most difficult for scenario developers to use.

### *c. Input and Output*

COMBATXXI has different tools to manage the data. The management consists of importing, exporting, and modification for the purpose of either working with a server, comparing scenarios, or simply for displaying the data. When needed, data is stored in Data Objects which fill themselves through structured query language (SQL) queries to the databases. These databases are implemented in different database management systems such as Microsoft Access and OpenOffice. Besides the database files, the COMBATXXI scenarios use configuration files, scenario XML files, communication configuration files, situational awareness files, behavior scripts, and environment files. For the output, the model events store information to different logger files based on the user choice at the start of scenario execution.

## 2. Hierarchical Task Networks

### *a. Basic HTN Overview*

HTN is a common problem-solving technique that abstracts away much of the problem's details by reducing the problem to a hierarchical structure (Erol et al., 1995). Specifically, it enables automated planning. Its original purpose was to bridge the gap between AI planning and operations research management and scheduling. In HTN planning, the world is described in states, and the state transitions are called actions or tasks (Erol et al., 1995). A task network consists of a collection of tasks to be performed. Each task has a name and a list of arguments that can be constant or variable, and it can be a primitive, compound, or goal task. A primitive task is executed directly while a compound task is a list of goal and primitive tasks. The goal task is a desired property in the field of study. The plan to solve a problem is considered achievable when all compound tasks are reduced to primitive ones and the goal state can be reached. The task network also has constraints on the tasks by either bounding the argument variables or imposing an order on the tasks. A specific branch of the HTN is executed only if the related constraint is met. Handling interactions among compound tasks is the most challenging part in HTNs because the planning could be undecidable when the subtasks

are interleaved or there is a recursion (Erol et al., 1994). Figure 3 shows the basic HTN planning algorithm.

1. Input a planning problem **P**.
2. If **P** contains only primitive tasks, then
   resolve the conflicts in **P** and return the result.
   If the conflicts cannot be resolved, return failure.
3. Choose a non-primitive task $t$ in **P**.
4. Choose an expansion for $t$.
5. Replace $t$ with the expansion.
6. Use critics to find the interactions among the tasks in **P**,
   and suggest ways to handle them.
7. Apply one of the ways suggested in step 6.
8. Go to step 2.

Figure 3. The basic HTN planning algorithm (from Erol et al., 1995)

***b. Example***

A hierarchical task network can produce a plan to mimic the behavior of some unit moving to its destination. The compound task would be "move to destination," and the goal task would be "Arrive at destination." The compound task could be decomposed into checking the availability of means of transportation, mounting the vehicle, driving to objective, and walking to objective. Since changing the plan whenever circumstances change is a human behavior, the HTN has to capture this. To account for changes, interrupt nodes were added to HTNs in the tasks where problems could occur. When an interrupt node is hit, the plan gets reevaluated from the beginning. For this example, problems could occur in the route. For instance, the vehicle could overheat or the terrain could be too difficult to navigate using a vehicle. The traversal of the HTN tree starts by checking the constraints "At destination," then "At base," and "transportation means available." Based on these constraints, the unit would either drive or walk, and then reevaluate the constraint "At destination." This procedure guarantees the unit does not get stuck in any task. Figure 4 shows the design of the HTN for this scenario.

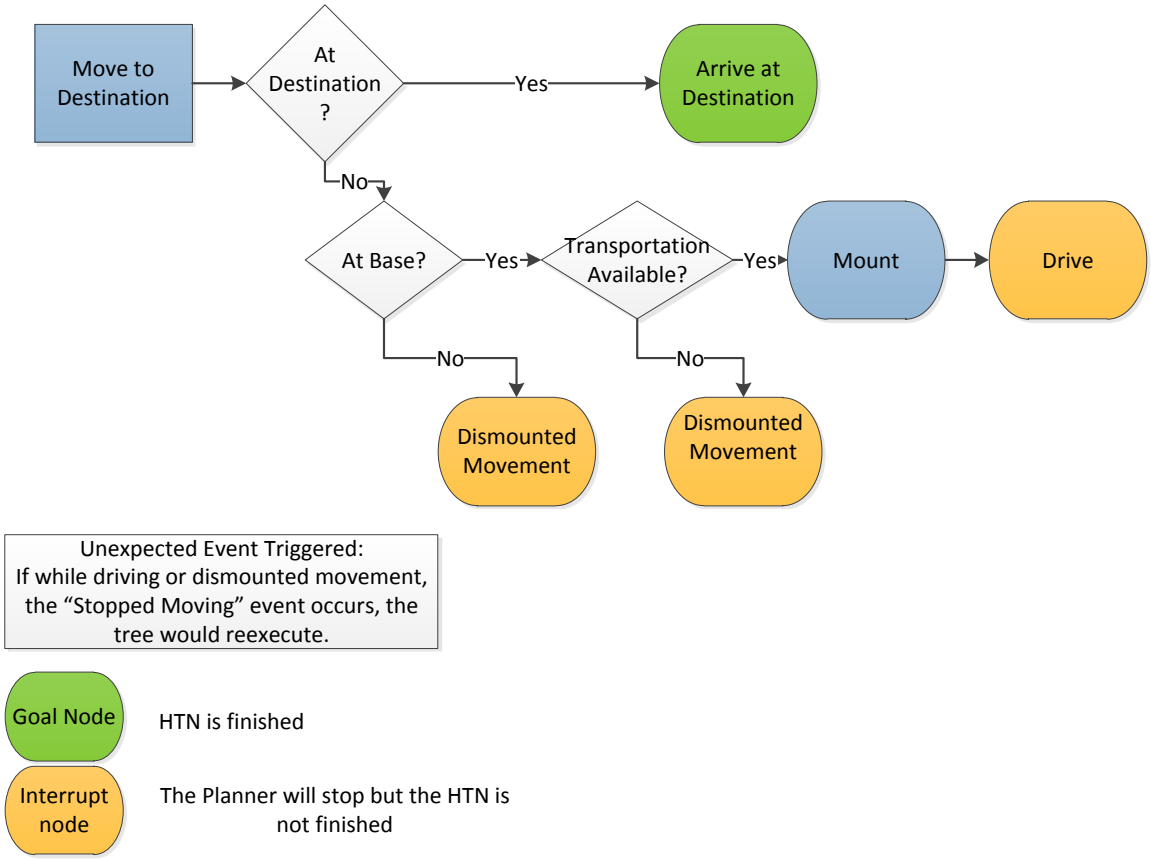


Figure 4. Movement-to-destination behavior in a HTN.

## B. DISCRETE EVENT SIMULATION

### 1. Definition

Fishman (1978, p. 1) defines a discrete event system as "one in which a phenomenon of interest changes value or state at discrete moments of time rather than continuously with time." He states that the "concepts, methods, and procedures for modeling the behavior of a discrete event system, for translating the model into code executable on a digital computer, and for analyzing sample sequences that emerge from the program's execution … comprise the topic of discrete event simulation." (Fishman, 1978, p. 1) Also, Law and Kelton (2000) define discrete-event simulation as "the modeling of a system as it evolves over time by a representation in which the state variables change instantaneously at separate points in time." They define event as "an instantaneous occurrence that may change the state of the system." (Law & Kelton, 2000,

p. 6)A Discrete Event Simulation (DES) model has states, events, and scheduling relationships between events. States have constant parameters and variables. The values of the variables describe the state of the simulation at a given point in time. The values of the variables remain constant for a certain period of time then change. This change is called state transition or an event. Thus, for any given event, there is at least one state variable that changes its value. Events are organized via a scheduling procedure which allows events to schedule each other. The time between the events can be fixed or can be variable. Figure 5 illustrates the common event processing cycle.

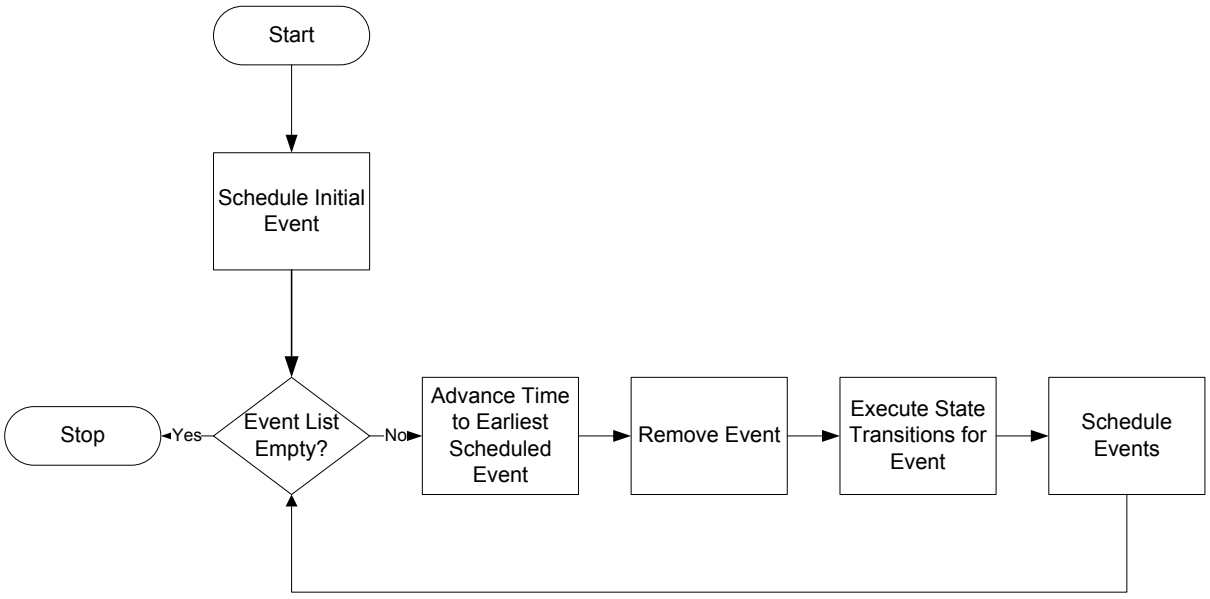


Figure 5. Next event algorithm (from Buss, 2014)

An initial event, which occurs at time 0.0, sets all state variables to initial values and schedules at least one other event. The event list keeps track of pending events in time order with at least their identifiers and scheduled times. The model stops when the event list is empty.

### 2. Example

A simple example of DES models an incident that happens periodically with a changing frequency of occurrence (see Figure 6). The state keeps track of the number of occurrences of the incident with a state variable N initialized to zero. The parameter is

used to generate the different times between the occurrences. The events are "Run" and "Arrival." The event "Run" schedules the event "Arrival" to occur after time "$t_A$." Finally, each event changes some state variables as shown under the events in Figure 6.

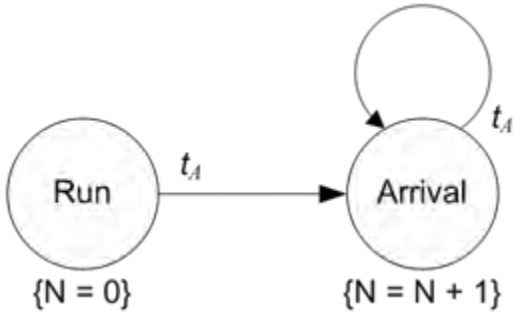


Figure 6. Single-incident event graph (from Buss, 2014)

## C. SEMANTIC WEB AND WEB SERVICES

### 1. Semantic Web

The use of semantic web (SW) concepts and technologies has solved many problems of information overload and time constraint in different domains, and has enabled information superiority (Childers, 2006).

The Extensible Markup Language (XML) provides a framework for describing, structuring, and exchanging data which can be used over a network. It is the W3C standard for creating vocabularies of a domain (Extensible Markup Language, 2014). XML documents must follow some rules to be parseable by XML parsers. First, tags come in pairs. For example, <name>Jack</name> is a well-formed expression in an XML document. Second, XML is hierarchical. This feature is important for human readability as well as for organization. However, the machine does not know the meaning of an XML document.

SW languages such as Resource Description Framework (RDF) and Web Ontology Language (OWL)are XML documents that promote semantic interoperability.

Figure 7 shows the different layers of the semantic web (Childers, 2006) in a depiction of increasing levels of semantic specificity.

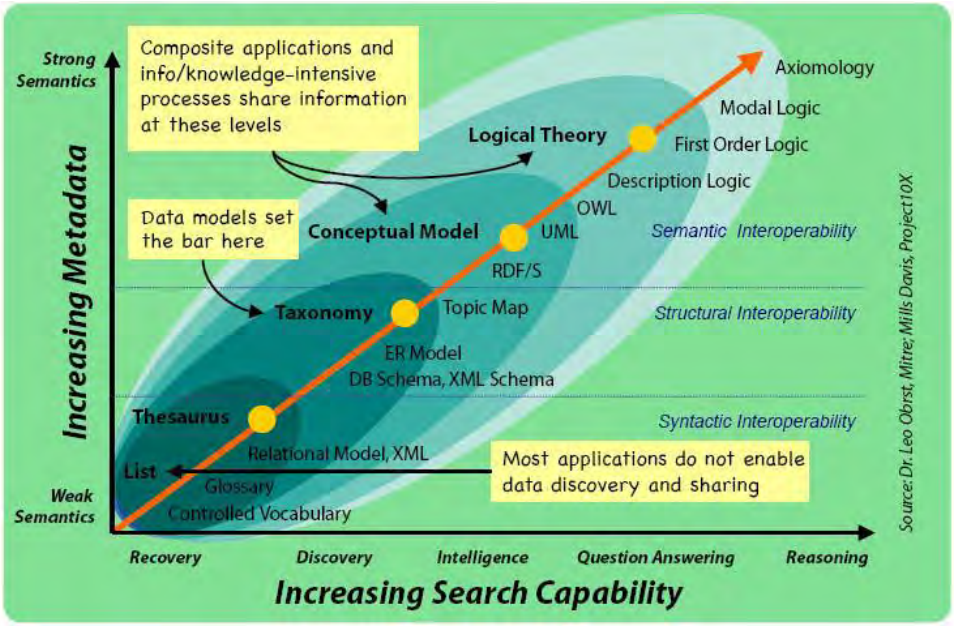


Figure 7. Illustration of the levels of interoperability required for machine understanding of data across systems (from Obrst, 2006).

RDF allows describing resources or adding metadata (Hjelm, 2001). A resource can be any object. Naming conflicts are resolved using the Uniform Resource Identifier. (URI) RDF is a W3C specification and it can be serialized in XML for interoperability. An RDF assertion is composed of a subject, a predicate, and an object. RDF is not hierarchical. Below is an example of an RDF document that describes a book by its title and owner (Childers, 2006).

```
<rdf :RDF
        xmlns:rdf = "http://www.w3.org/1999/02/22-RDFSyntax-ns#"
        xmlns:ex = "http:// www.resources.org/"
        xmlns:dc= "http://purl.org/dc/elements/1.1">
        <rdf :Description rdf :about = "ex:book">
                <dc:title>Practical RDF</dc:title>
                <dc:creator> Shelley Powers </dc:creator>
        </rdf:Description>
</rdf:RDF>
```

RDF can be extended to RDF Schema to describe whole concepts instead of specific resources. The RDF Schema brings an object-oriented feature by defining classes and properties which enables the creation of ontologies. However, it is still not expressive enough to show constraints and logic (Passin, 2004).

OWL is the next layer up in the SW representation. It is the standard for creating ontologies that can be serialized as RDF/XML for interoperability. In addition, there is an OWL Application Program Interface (API) for manipulating and querying ontologies. Another way to explore ontologies is to use other languages, such as the Semantic Web Rule Language (SWRL) and Sparql Protocol and RDF Query Language (SPARQL), to query and reason about classes, properties, and instances. SWRL is a combination of OWL-DL and the Rule Markup Language. The following example shows a SWRL rule that finds all instances of "HostileContact" with speed greater than five and sets their "isThreat" property to true (Childers, 2006).

HostileContact(?x) ∧ speed(?x, ?speed) ∧
swrlb:greaterThanOrEqual(?speed, 5)→ isThreat(?x, true)

Some of the limitations that SWRL faces are its difficult syntax, predicate restriction to the Boolean type, and lack of support of SW reasoners. This latter problem was reduced using new rule engines for SWRL while keeping the DL reasoners for reasoning with OWL constructs (Childers, 2006).

SPARQL is used for querying RDF expressions. Two common tools for building SW applications that use SPARQL are Jena and Twinkle. The following example shows a simple syntax of the SPARQL language that queries all subjects with speed of 10 and course equal to 45(Childers, 2006).

```
PREFIX taml: <http://usw.xml.wg/>
SELECT ?x ?targetSpeed ?targetCourse
FROM <file:TAMLExample.rdf>
WHERE
{
        ?x taml:Speed ?targetSpeed.
        ?x taml:Course ?targetCourse.
        FILTER (?targetSpeed = "10" && ?targetCourse= "45.0")
}
```

## 2. Web Services

Web services are technologies that enable interoperability among many applications regardless of the platform and the language (Chappel et al., 2002). They are composed of three major elements: Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL), and Universal Description, Discovery, and Integration (UDDI). SOAP prepares the XML documents and encodes Remote Procedure Calls (RPC) for transport. Particularly SOAP Version 1.2 "provides the definition of the XML-based information which can be used for exchanging structured and typed information between peers in a decentralized, distributed environment." (Mitra, 2007) WSDL describes the information needed to communicate with a server. More specifically, it gives the answers to the questions: Who? What? Where? Why? How? UDDI permits clients to easily find web services. SOAP messages are sent over Internet protocols (David et al., 2002). Figure 8 shows a simple web service architecture.

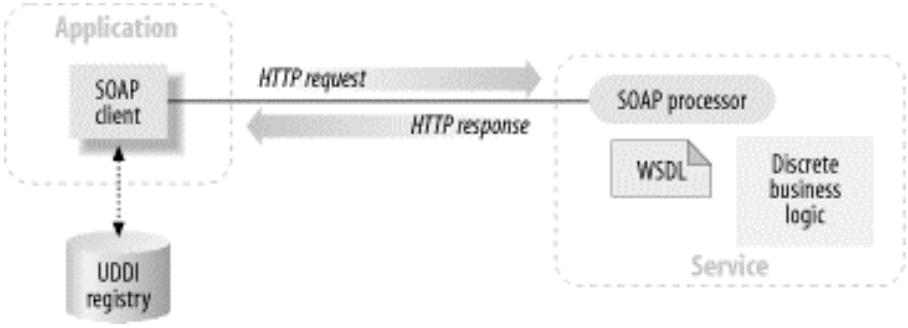


Figure 8. Simple web service interaction (from David et al., 2002)

The Mobility Common Operational Picture (M-COP) project represents a good example of the use of web services in modeling and simulation in DOD. Richmond et al. examined in detail how ground mobility information can be represented, and looked at the Web Services that would support the M-COP. The M-COP data model provided a core set of information requirements which included raw data and logic products to support movement planning for ground mobility. The different categories of information requirements included, but were not limited to: terrain, obstacles, weather, maneuver analysis, forces, and threat analysis. The connection between the M-COP model and

other systems was envisioned to be through web services using Machine-to-Machine Messaging that takes the form of SOAP over HTTP. The model was also open to other interoperability techniques. The Joint Consultation Command and Control Information Exchange Data Model (JC3IEDM) is a common data model for interchange of operations data across different systems in multi-national coalitions. M-COP planned to map its information model to that of JC3IEDM for wider application.

## D. ONTOLOGY

### 1. Definition

An ontology is an "explicit formal specification of the terms in a domain and relations among them," (Gruber, 1993) and are widely used. Some of their purposes are sharing common structure of information, reusing and analyzing domain knowledge, and separating domain knowledge from operational knowledge. An ontology is composed of classes (concepts), slots (properties), and facets (restrictions). An ontology plus the instances of its classes constitute the knowledge base (Noy et al., 2001).

### 2. Ontology Development

In practical terms, first we determine the scope of the ontology. Second, we enumerate all possible terms in the ontology without worrying about the structure of the ontology. Then, we define the classes, subclasses, and super-classes, and we define the slots from the list of terms constructed above. A slot should be attached to the most general class having the property. Next, we describe the allowed values for the properties. Facets describe the type of the value, the cardinality, and the domain and the range if the type is an instance. Finally, we create the instances (Noy et al., 2001).

The class hierarchy should be checked using the "is-a" and "kind-of" relations. Generally, a class has between two and twelve subclasses. Multiple-inheritance is allowed, but class cycles are not allowed. To introduce a new subclass instead of a property, the new subclass should have additional properties, additional restrictions, or participates in a different relationship. Another perspective that helps distinguishing between new classes or new properties is to evaluate the importance and the implications

of the term in the ontology. For example, the way an improvised explosive device (IED) is packaged could be a property of an IED named package_type if it does not have any implications for the other objects. However, if this concept is important, then we need to create three disjoint subclasses named packaged_IED, suicide_Bomber_IED, and vehicle_Borne_IED based on the delivery method (Teters, 2013).

Sometimes, it is useful to have an inverse of a property without entering the values in both slots. The use of inverse slots allows the system to automatically enter the values. It is useful also to declare a default value for the slots unless the user changes it. Finally, it is common to capitalize class names, use prefix has- or suffix –of for slot names (Noy et al. 2001).

## E. KNOWLEDGE-BASED SYSTEMS

### 1. Introduction

Many old systems placed emphasis only on heuristics to solve problems, but the new design paradigm for intelligent systems stressed the need for expert knowledge and knowledge handling facilities (Mylopoulos et al.,1983). Before going into the details of this domain and defining its concepts, an informal organization chart is provided in Figure 9 to give a general overview of the domain. It is a non-exhaustive and informal categorization. Categories might not be disjoint as in the case of types of reasoning or subclasses of KB systems.

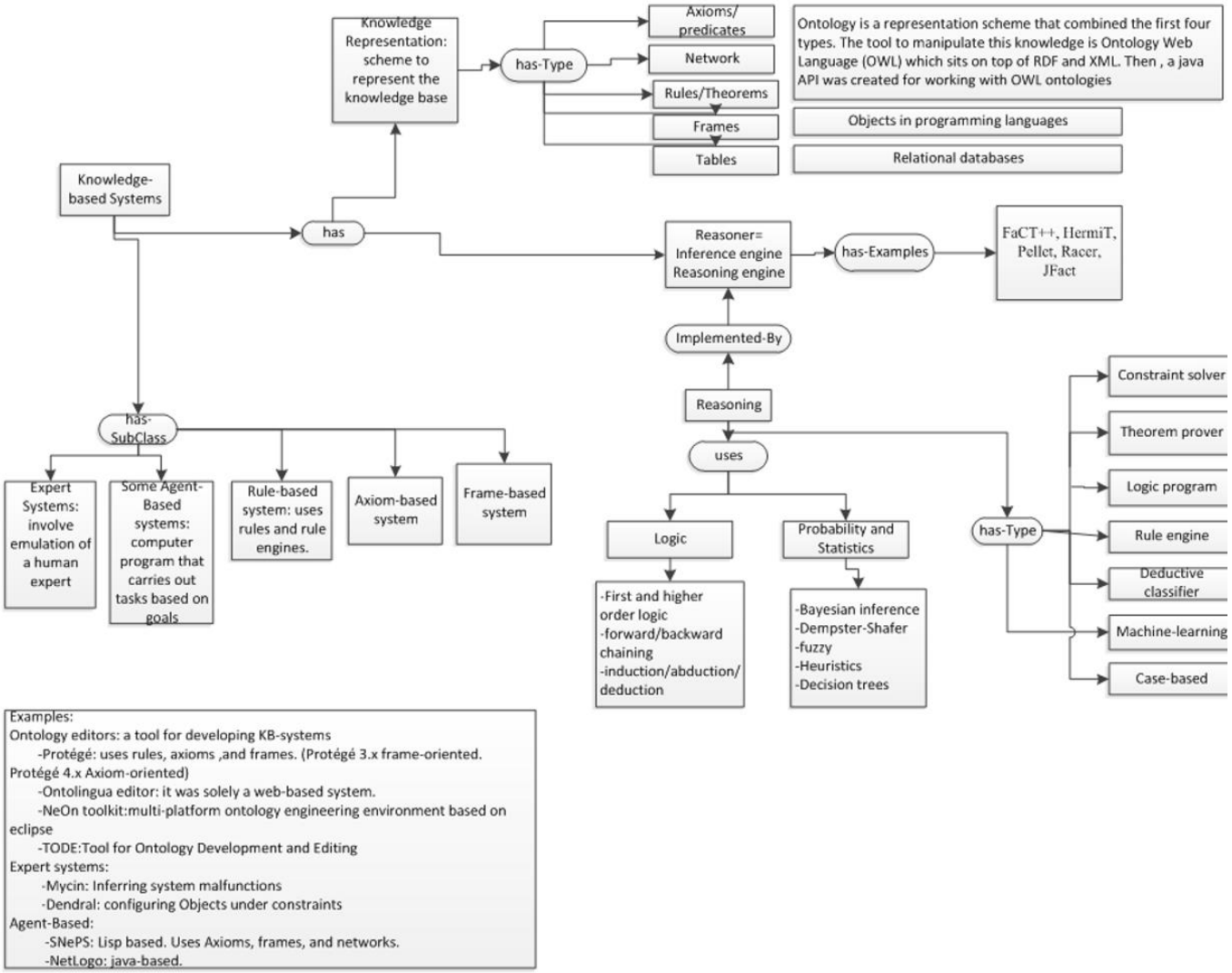


Figure 9. KB-systems organization chart

Knowledge is a relation between a knower and an abstract entity that is either true or false (Luger, 2005).This abstract entity is called a proposition. If the knower knows the proposition, we take it to be true. Representation is a relationship between the problem domain and computational domain where the second takes the place of the first. The outcomes of inference systems are considered either observations or possible actions (Luger, 2005). The second one is the *representor* that is often a symbol from an alphabet. Therefore, knowledge representation is the field of study concerned with using symbols to represent propositions. Reasoning is a form of computation that manipulates the symbols to produce (infer) more propositions from explicit ones. However, it is difficult to decide which proposition to use. It can lead to incomplete logic when an important proposition is left out, or it can even lead to unsound logic when incorrect answers are returned (Brachman & Lavesque, 2003). Therefore, there have been several attempts to provide sufficiently precise notations to represent knowledge and then come up with different reasoning techniques. All the resulting propositions are considered to be the knowledge base that models the universe of interest. The notation used is called the representation scheme. There are declarative and procedural schemes (Mylopoulos, 1980).

## 2. Representation Schemes and Reasoning

### *a. Logical Representation Schemes*

In this scheme, the knowledge base consists of logical formulas as the atomic units, and they can be added or deleted. It generally uses constants, variables, predicates, and quantifiers in a first order or higher order logic. The simplicity of first order logic makes it a good first step to study knowledge representation and reasoning.

In creating a knowledge base, it is a good idea to start with the individuals; for example, Jack, Mary, BMW, and Burger King. Next, we define and limit the types of individuals as objects; for example, Man, Car, and Restaurant. Then we define the attributes that the individuals can have such as Happy, Fast, and Closed. Using these three components, we can construct unary predicates such as Man (Jack), Car (BMW), Restaurant (Burger King), Happy (Jack), Fast (BMW), and Closed (Burger King). To

construct n-ary predicates, we introduce relationships such as FriendOf, MadeIn, and Purchased. They are used as follows: FriendOf (Jack, Mary), MadeIn (BMW, Germany), and Purchased (BMW, Jack). All these predicates are called basic facts, and can be implemented using relational databases. There are also Complex Facts which include the basic facts plus quantifiers and connectives such as: ∃ x (Happy(x) ΛPurchased (BMW, x)). The previous complex fact means that there is someone who is happy and has purchased a BMW. Basic and complex facts are sufficient to represent the world. However, they do not guarantee an error-free representation. Therefore, another type of facts was introduced. Terminological facts include disjointness, subtypes, exhaustiveness, symmetry, inverses, type restrictions, and full definitions. Finally, it is possible that we cannot determine the details and the properties of an individual in advance. To address this issue, it is useful to introduce new abstract individuals to link the concrete individuals to any new property that we want to add. For example, if we want to add the date Jack bought the BMW and the price. It is not efficient to search for previous predicates and change them (i.e., change Purchase (BMW, Jack) to Purchase (BMW, Jack, 2010, $30000)). However, new rules should be added as follows: action (p) ^ agent (p, Jack) ^ object (p, BMW) ^ time (p, 2010) ^ price (p, $30000). This process is called *reification*.

This scheme is characterized by simplicity and economy of notation, availability of formal semantics, and easiness of information retrieval and constraint checks. However, it is not scalable because of lack of organizational principles, and it does not fully represent procedural and heuristic knowledge. Moreover, to determine that a proposition is a logical consequence of others is generally unsolvable. This is known as the fundamental computational intractability of first-order entailment (Brachman & Lavesque, 2003). These advantages and disadvantages can be summarized in expressiveness and efficiency. Since efficiency is as important as expressiveness, there needs to be a trade-off between them to achieve an optimized representation and reasoning scheme (Luger, 2005). Some of the techniques of reasoning are resolution, induction, approximation, and horn clauses. The idea of horn clauses, for example, was to limit the scope to only some subsets of first-order logic. This solved the computation

issue while limiting the scope. The question then became whether the new scope was expressive enough for the respective problem domain (Brachman & Lavesque, 2003).

### *b. Network Representation Schemes*

In an attempt to better organize the knowledge base, the universe of discourse has been described in terms of objects (nodes) and associations (edges). Thus, the knowledge base became a directed graph. This scheme is called semantic networks or conceptual graphs. Data manipulation is achieved through four basic operations: deletion, union, insertion, and simplification. Moreover, the reasoning algorithms follow the links and use graph theory to retrieve information and solve problems. The basic application of semantic nets is to represent a fact or an action like the one described in the first order logic. Figure 10 shows the fact that Jack bought a BMW. More advanced applications in this scheme dealt with natural language processing; defining a word means traversing the network until the meaning becomes understood. Finding the relationship between two words means finding a link through a common node (Luger, 2005).

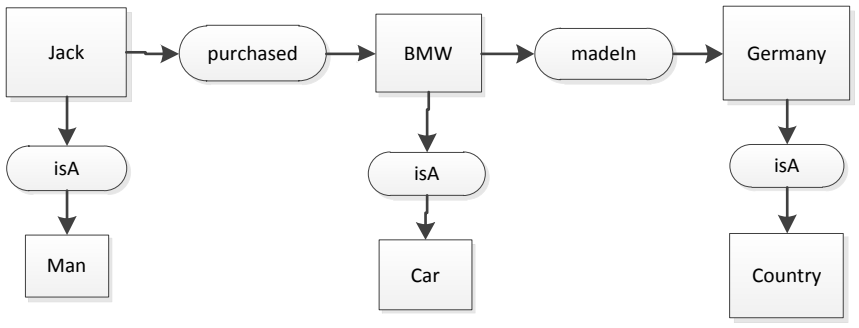


Figure 10. Semantic network of the operation made by Jack

Some of the advantages of this scheme are the easiness of information retrieval, organization, and visual representation. The downside of this technique is the lack of formal semantics and standard terminology (Mylopoulos, 1983), even though Simmons (Simmons, 1973) addressed this issue by using the structure of the English language to

limit the relationships to agent, action, object, instrument, location, and time. This list was further expanded later by Schank and Rieger in 1974 (Luger, 2005).

### *c. Procedural Representation Schemes*

In this scheme, the knowledge base is a collection of procedures of some language with the clauses divided into facts and rules. The schemes belonging to this class can be differentiated based on their activation mechanisms and control structures. For example, in PLANNER scheme the knowledge base is a collection of assertions and theorems. The theorems are executed in the backtracking technique based on searches or modifications of the knowledge base (Mylopoulos, 1983). Production systems are another type of this scheme. They are rule-based systems that use either data-driven reasoning or goal-driven reasoning. They follow the forward-chaining computation. They have a working memory (volatile database). The working memory is a tuple of the form: (type attribute: specification). For example, (person name: Jack age: >25 curriculum: computerScience).The rules are usually written in the form: if conditions then actions. The production system operates in a cycle of three elements, and it halts when no rules are applicable to the working memory. First, it finds the applicable rules. Second, it chooses which rules to fire. Finally, it performs the actions of the fired rules. The actions can add, remove, or modify the working memory. For example, IF (student name: Jack) THEN ADD (person name: Jack). Production systems were successful in solving many practical problems through their modularity, fine-grained control, and transparency (Brachman & Lavesque, 2003).

Like resolution, automated theorem-proving is an inference procedure that tries all possibilities in the knowledge base to reason about the truth of a given proposition. This is inefficient and sometimes infeasible. In addition, sometimes we want to control the reasoning process via providing more information specific to the situation or limiting the scope of reasoning.

### *d. Frame-Based Representation Schemes*

In all previous representation schemes, every piece of information was independent. This caused the knowledge about any object to be scattered across the knowledge base, which is inefficient. Object-oriented representation is an approach to help create a structured and organized knowledge base. In 1975, Minsky used the term frame as the data structure representing a stereotypical situation. The frame has slots for the objects that play a role in the scene and for the relationships amongst them. There are two types: individual frames represent a single object, and generic frames represent a class of objects. Therefore, the individual frame is an instance of the generic one. A frame is a list of slots. Notice that this is similar to the working memory of the production systems. Much of the reasoning in the frame system is done through inheritance of properties and procedures. It is done in three steps. First, the user instantiates some generic frame. Second, the new instance fills in its slots via inheritance. Third, the inherited procedures run, and possibly instantiate new instances. Extensions to the frame system included the use of multiple slot fillers, slot facets, and meta-frames. Finally, even though object-oriented programming and frame-based representations were developed concurrently sharing many features, they differ in the fact that frame-based systems tend to be more centralized and work in a cycle whereas object-oriented programming has independent objects sending each other messages (Brachman & Lavesque, 2003).

## 3. Examples of Reasoners

As seen in Figure 9, reasoning takes different forms such as constraint solving, deductive classification, and theorem proving. Reasoners also use different techniques such as description logic and non-monotonic logic. Therefore, many reasoners have been implemented. The RacerPro reasoner provides consistency checking, classification, and OWL Query Language (OWL-QL) resolution (Abburu, 2012). Pellet can check ontology consistency and classify taxonomies, and it is able to connect with Protégé and Jena. Unlike RacerPro, Pellet is an open source java-based reasoned (Sirin, 2005).Finally, HermiT fully supports OWL2, and is the fastest reasoner in classifying complex ontologies. HermiT is an open-source java library, and it can handle ontologies according

to the OWL API (Horrocks, 2013). Tables 1 and 2 provide more information on comparative capabilities and performance. The unit under the reasoners is "seconds."

Table 1. Reasoners performance evaluation (after Horrocks, 2013)

| OntologyName | Classes | Properties | HermiT | Pellet | FaCT++ |
|---|---|---|---|---|---|
| EMap(Feb09) | 13737 | 2 | 1.1 | 0.4 | 34.2 |
| GOTermDB(Feb06) | 20526 | 1 | 1.3 | 1.3 | 6.1 |
| DLPExtDnS397 | 96 | 186 | 1.3 | timeout | 0.05 |
| Biological Process(Feb09) | 16303 | 5 | 1.8 | 4.0 | 8.0 |
| MGEDOntology | 229 | 104 | 2.1 | 19.6 | 0.04 |
| NCIThesaurus(Feb09) | 70576 | 189 | 58.2 | 12.3 | 4.4 |
| OBI(Mar10) | 2638 | 83 | 150.0 | timeout | 17.2 |
| FMALite(Feb09) | 75145 | 3 | 211.1 | timeout | timeout |
| FMA-constitutional part | 41648 | 168 | 1638.3 | timeout | 396.9 |
| GALEN-doctored | 2748 | 413 | 1.8 | timeout | 2.5 |
| GALEN-undoctored | 2748 | 413 | 6.7 | outofmem. | 11.6 |
| GALEN-module1 | 6362 | 162 | outofmem. | timeout | timeout |
| GALEN-full | 23136 | 950 | outofmem. | timeout | timeout |

Table 2. Reasoners compatibility and support (after Abburu, 2012)

| | **Pellet** | **RACER** | **FACT++** | **HermiT** |
|---|---|---|---|---|
| **Methodology** | Tableaubased | Tableau based | Tableaubased | Hypertableau based |
| **Soundness** | Yes | Yes | Yes | Yes |
| **Completeness** | Yes | Yes | Yes | Yes |
| **Expressivity** | SROIQ(D) | SHIQ | SROIQ(D | SROIQ(D) |
| **NativeProfile** | DL,EL | DL | DL | DL |
| **RuleSupport** | Yes(SWRL) | Yes(SWRL) | No | Yes(SWRL) |
| **OWL API** | Yes | Yes | Yes | Yes |
| **OWLLinkAPI** | Yes | Yes | Yes | Yes |
| **ProtégéSupport** | Yes | Yes | Yes | Yes |
| **NeOnSupport** | Yes | No | No | Yes |
| **Jena Support** | Yes | No | No | No |
| **Impl.Language** | Java | LISP | C++ | Java |
| **Availability** | Opensource | Commercial | OpenSource | Open source |

## F. PROTÉGÉ

### 1. Introduction

Protégé was developed in 1987 as a small application for knowledge acquisition in knowledge-based systems. It incorporates Open Knowledge Base Connectivity (OKBC) model, relational databases, XML, RDF, and OWL. All knowledge representation schemes mentioned earlier can coexist in one system. That is, a piece of software can incorporate and implement some of the capabilities of these schemes. For example, Protégé is mainly frame-oriented, but it also uses the logical and the network representation schemes by adding plugins for reasoners and graphical visualization. In addition, in later versions, Protégé became mainly axiom-oriented which is based on the procedural representation.

### 2. Evolution of Protégé

#### *a. Protégé-I*

Expert systems were based on the idea of a central inference engine that knowledge engineers can use with different knowledge bases resulting in different expert systems. The knowledge engineer had to become familiar with the problem domain such as the concepts and the reasoning strategies, and then he or she had to formalize the domain in a generic way. However, the domain expert was only a source of knowledge in the beginning and during testing. The construction of knowledge bases by the knowledge engineers was a difficult and time-consuming task, besides the possible errors that can occur because of misunderstanding between the knowledge engineer and the domain expert. The original Protégé aimed at solving this issue by allowing the domain expert to construct the knowledge base. The expert system had three components: the knowledge engineer provides the structure by building the knowledge acquisition tool (for example, forms with widgets (text fields, check-boxes) to enter information), the domain expert instantiates the domain concepts through the knowledge acquisition tool, and finally the end user interacts with the system for decision support. The first component is further achieved through Protégé. The methodology is that knowledge acquired in a stage is the meta-knowledge for the following stage, as shown in Figure 11 (Gennari et al.,2003).

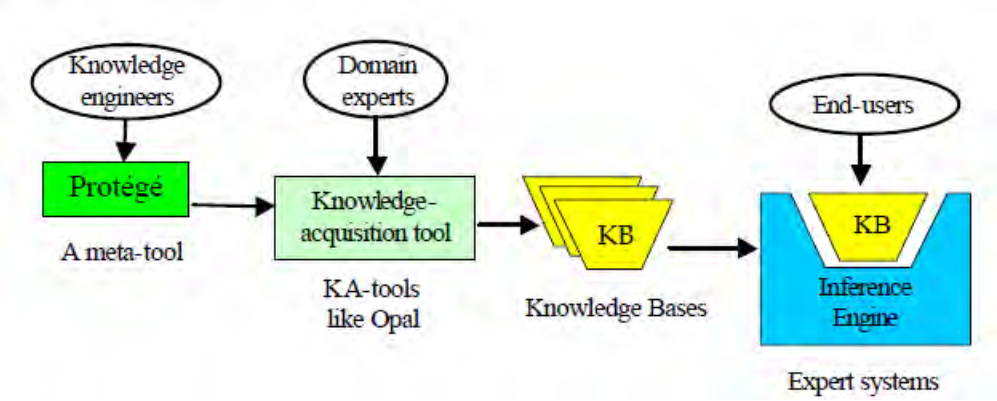


Figure 11. The use of Protégé in expert systems (from Gennari et al., 2003)

Protégé-I was limited to one inference engine, therefore the system was only well-suited for some specific applications.

### *b. Protégé-II*

The main goal of this version was to make the problem-solving methods independent from the knowledge base, make each component of the system reusable, and create the mapping between the knowledge base and the problem-solving method. In addition, the notion of ontologies was introduced, and ontologies became the basis for the knowledge acquisition tools. They used a frame-based formalism which consists of classes, instances, slots, and facets. There are three approaches in building an ontology: domain (focus on the concepts of a domain), method (focuses on the requirements through input and output of methods), and application (specific implementation).

Protégé-II has four subcomponents: Maitre (building *ontologies*), Dash (manipulating default layout of the knowledge acquisition *KA-tools*), mediator (used by domain experts to build and edit *knowledge bases*), and marble (mapping interpreter). The downside of Protégé-II is that it is difficult to change the ontology after building the knowledge base (Gennari et al., 2003).

### *c. Protégé/Win*

The purpose of this version was to make Protégé operable under the Windows operating system, allow the inclusion of ontologies, integrate the subcomponents under

one software, and improve the knowledge acquisition tools. The latter improvement allowed the generation of multiple KA-tools with different views and formats from a single ontology (Gennari et al., 2003).

### *d. Protégé-2000*

By popular demand, Protégé had to fix two issues: the limitation of changing the ontology after building the knowledge base from Protégé-II and domain-specific adaptation. The first problem was solved by basing the knowledge model on the OKBC which allowed classes and instance to be treated the same. In particular, it allowed the creation of meta-classes (templates). The second problem was solved by moving toward a plug-in architecture. The plug-ins allow users to customize and build a domain-specific interface through the addition and removal of tabs. Developers might also choose to build an entire application to control users' interaction with the knowledge base through calls to the knowledge model API. There are also backend plug-ins if the user chooses a different storage format. Currently, the system uses a special-purpose file format, RDF files, XML files, and relational database format. The resultant architecture is shown in Figure 12 (Gennari et al., 2003).

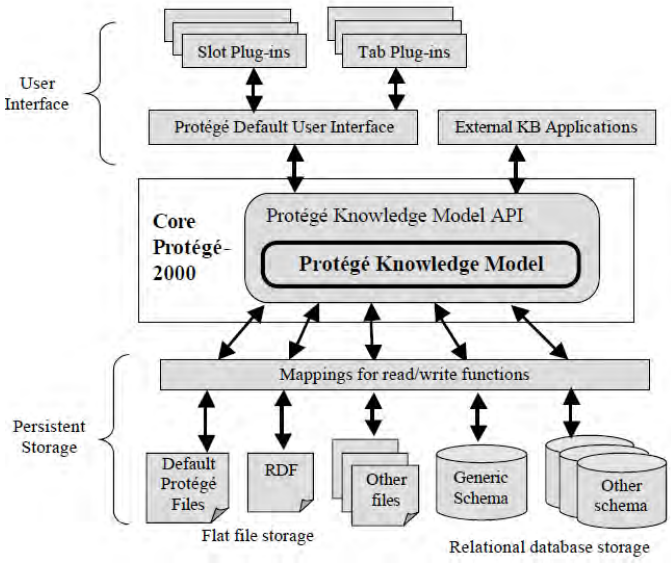


Figure 12. Structure of Protégé 2000(from Gennari et al., 2003)

### *e. Protégé y.x*

Protégé was released to public in various versions depending on the underlying technologies, enhancements, or bug fixes. There were Protégé 0.x, 1.x, 2.x, 3.x, 4.x, and Web Protégé. However, it is only worth mentioning Protégé 3 and 4 because of their major changes and stability. Protégé 4 is not an improvement of Protégé 3. Protégé 4 follows a totally different approach. Until now, they are both still used depending on the application. To work with frames, RDF, or OWL1.0,Protégé 3 is recommended whereas Protégé 4 is recommended for working with OWL2.0. In addition, in Protégé 3, direct access can be achieved by the Protégé-OWL API which is built on top of the frame-based system while Protégé 4 is built on top of the OWL API. Finally, Protégé 4 is optimized for large ontologies ("Choosing between versions of desktop Protégé," 2013).

### *f. An Example of Ontology Creation in Protégé*

Below are some steps to follow when creating an ontology. For more details, see the practical guide to building OWL ontologies (Horridge, 2011):

- Create classes and subclasses
- Make them disjoint
- Add object properties to describe the relationship between two instances of a class
- Add data properties to describe the relationship between instances and data values
- Add axioms to classes (equivalent or subclasses)
- Add a closure axiom (because an absent property does not mean it is false)
- Add a covering axiom (any subclass should have a type)
- Create instances
- Perform object and data properties assertions
- Run queries to retrieve more information than what is explicitly entered.

Figures 13, 14, and 15 show snapshots of the creation of an ontology that models a university consisting of teachers, students, and courses using some of the steps aforementioned.

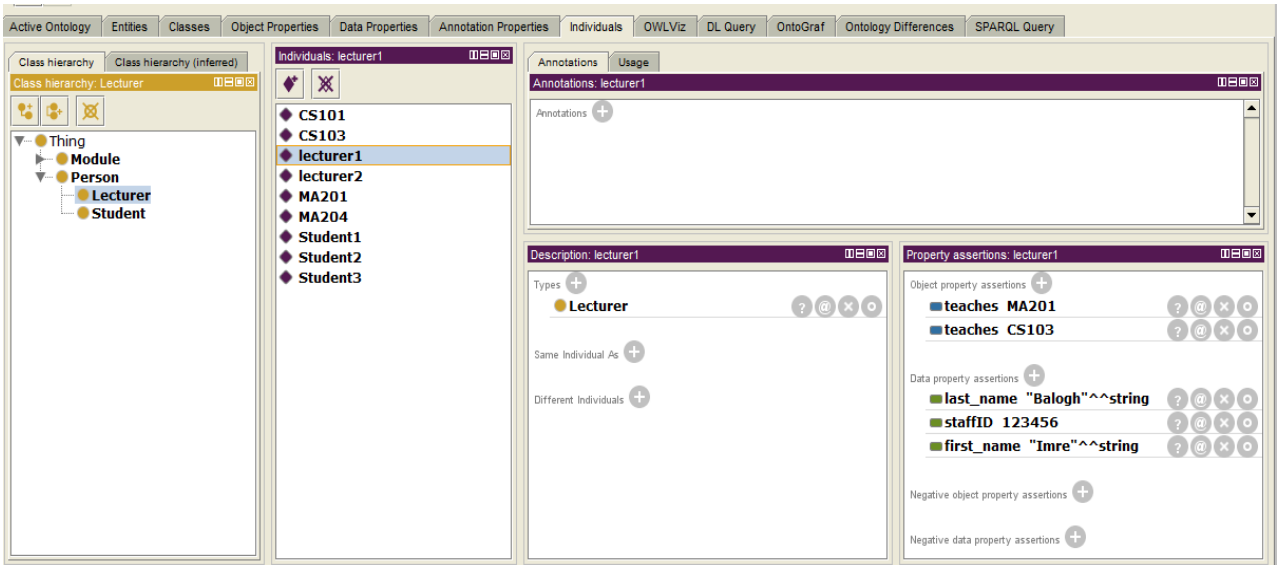


Figure 13. Creating an Ontology in Protégé

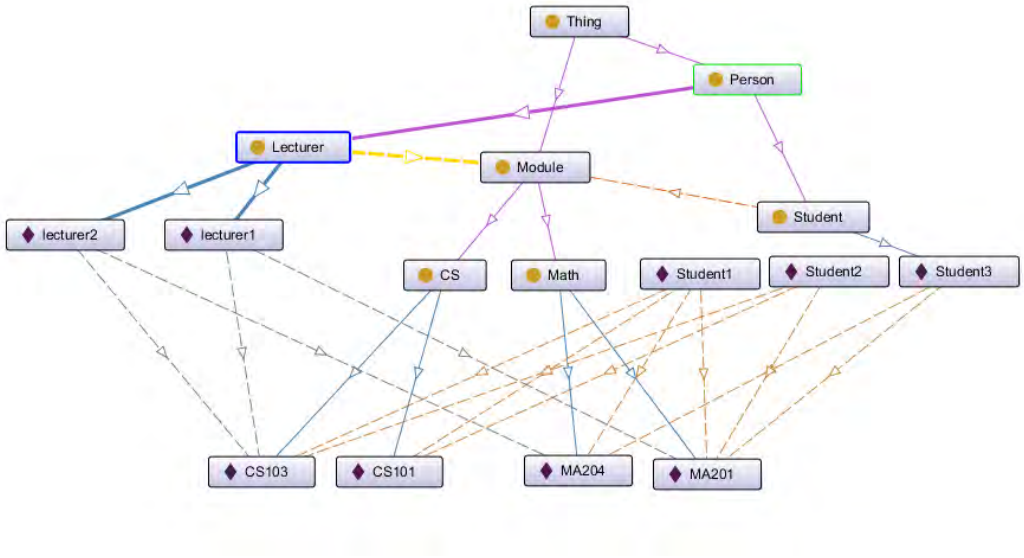


Figure 14. Visualization of the university ontology in Protégé

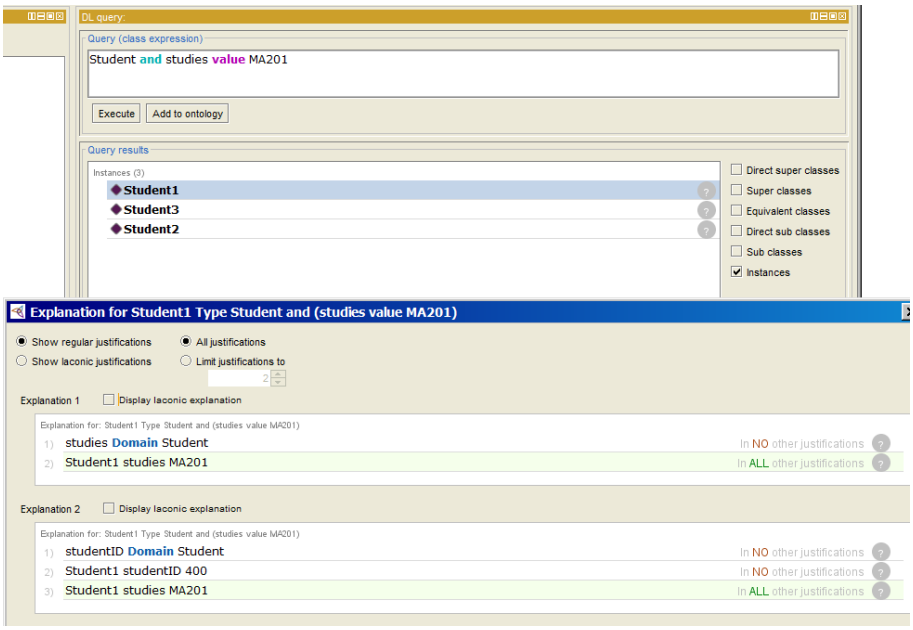


Figure 15. Using the reasoner to query some information

# III. METHODOLOGY

To reiterate, the research question is to investigate the use of automated reasoning in COMBATXXI entities' behavior and their decision-making capabilities. This can be achieved by connecting a knowledge representation system with inference capability to COMBATXXI.As mentioned in Chapter I, this task can be divided into several subtasks. First, we have to determine a suitable knowledge representation system and a reasoner. Second, we need to choose a type of reasoning and the area of COMBATXXI to be tested. Third, we have to determine an appropriate means of communication with COMBATXXI. Finally, an evaluation of the results is also needed.

## A. POSSIBLE KNOWLEDGE REPRESENTATION AND REASONING SOFTWARE FOR INTEGRATION

There are many options of knowledge representation systems, reasoners, and connection schemes as seen in Chapter II. First, knowledge representation and reasoning can be done separately, and then the information can be serialized using the already established Web Services technologies. Second, work can be done using Jena which is a Semantic Web framework for Java, with querying using SPARQL as shown in Chapter II. The Web Services and Semantic Web approaches would be more effective if we wanted to have knowledge representation and reasoning in a different machine than COMBATXXI. However, for this problem, all the work can be achieved on one machine. Therefore, a direct approach is more suitable by linking the two codebases through their APIs.

## B. CHOSEN APPROACH

In this application, we aim to model memory and brain of a COMBATXXI entity in a specific scenario. To scope it more specifically, we look at the classification capability of an entity. Therefore, looking at the different knowledge representation schemes in Chapter II, the frame-based and axiom-based schemes seem the most promising, because for the classification capability we want to put the information in frames and describe it in statements. Looking at the types of reasoning, the deductive

classifiers seem the most promising, because they help find more hidden classes for the information. Therefore, since Protégé incorporates many representation and reasoning capabilities such as the aforementioned, it is an appropriate knowledge representation and reasoning system that can model the knowledge structure, knowledge base (memory), and the thinking of entities. Protégé 3.x is built on top of OWL-Protégé while Protégé 4.x is built on top of the OWL API. The only compatible reasoners with Protégé 3.x are pellet and SWRL-IQ.The compatible reasoners with Protégé 4.x are Pellet, Fact++, Racer, JFact, and Hermit (Abburu, 2012). Since Java is used in COMBATXXI and in Protégé, it is convenient to also use it to connect the two systems. Therefore, the method chosen is to build ontologies in Protégé, use the OWL-Protégé or the OWL API, and use a reasoner to manipulate and query the ontologies based on entity situation, as part of behaviors represented as HTNs in COMBATXXI. Specifically, when an entity reaches a node in the HTN that requires collecting data about some concept, it calls and passes data as arguments to a Java method. Next, this method manipulates and queries the ontology. Finally, this method sends the results back to the entity's behavior logic. Figure 16 shows how Java is used to link COMBATXXI and Protégé. The figure will be explained in details in the implementation section.

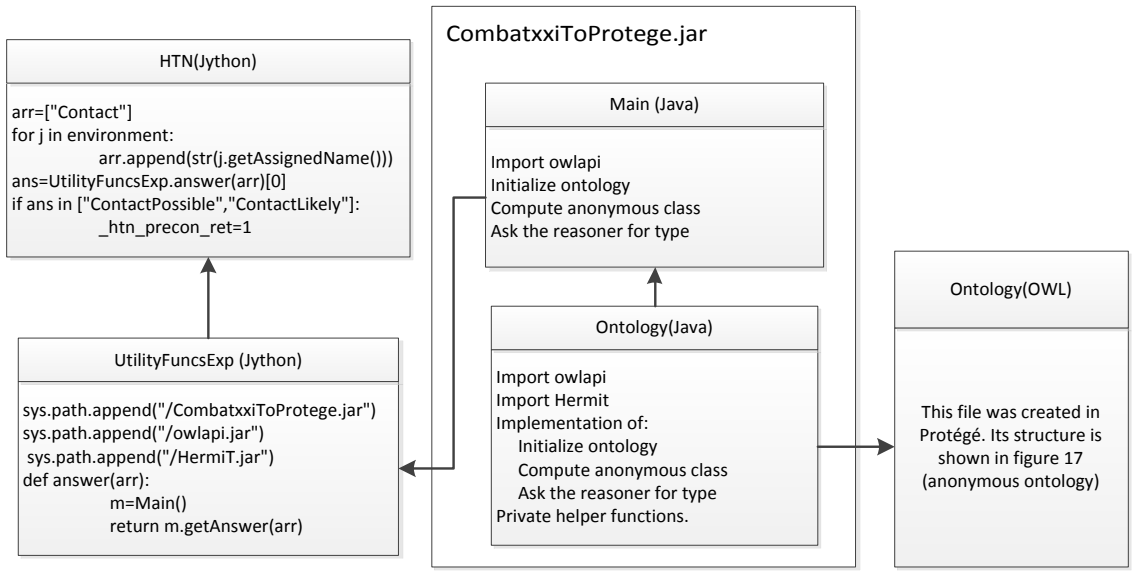


Figure 16. Process Diagram: COMBATXXI-Protégé Link

# IV. IMPLEMENTATION AND ANALYSIS

## A. IMPLEMENTATION

The implementation task consists of three subtasks. First, we need to set up a behavior in COMBATXXI that takes advantage of Protégé's reasoning capabilities. Second, we need to construct an ontology using Protégé that acts as the knowledge base. Third, we need to link the two systems. The research question was determining the feasibility of using a knowledge representation system with inference capability in a closed form combat simulation. Testing boundary conditions or providing a stress test is outside the scope of this thesis. However, for the implementation, two examples were selected to prove the feasibility. First, an entity wants to classify an IED based on information obtained from the battle space. Second, an entity decides on a type of movement based on the level of threat.

### 1. COMBATXXI

Suppose an entity in COMBATXXI collects some information, such as detection of a man, truck, cellphone, trash, and disturbed Soil, and wants to determine if there is an IED in the vicinity and what kind of IED it could be, or it wants to know the level of threat in some area and decide what tactical action should be taken. The COMBATXXI Behavior Studio was used to specify the behavior. First, the entity adds any useful information from the environment in a list. For simplicity, the names of buildings were used to mimic any objects we want to add to the environment. Then it sends that information to a python file (here named UtilityFuncsExp.py) that links to the JAR files which will do all the work and return an answer. The following python code from UtilityFuncsExp.py file shows how this file links the Java project and the COMBATXXI HTN:

```
import os, sys
sys.path.append(os.path.join(os.path.dirname(__file__), "dist/CombatxxiToProtege"))
sys.path.append(os.path.join(os.path.dirname(__file__),
"dist/lib/org.semanticweb.owl.owlapi.jar"))
sys.path.append(os.path.join(os.path.dirname(__file__), "dist/lib/org.semanticweb.HermiT.jar"))
```

```
from owlAPI import Main
def answer(arr):
        print arr
        m=Main()
        return m.getAnswer(arr)
```

Figures 17 and 18 show the HTNsspecifying the behavior for determining the IED and the level of threat.

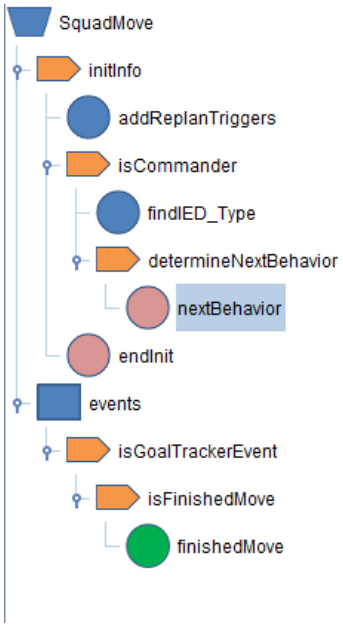


Figure 17. HTN: Finding the IED type.

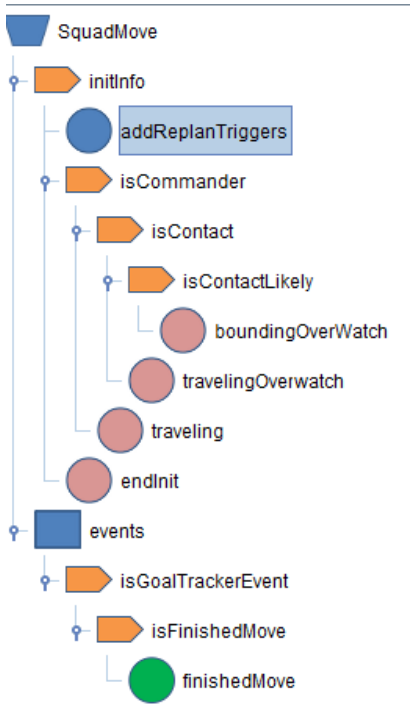


Figure 18. HTN: movement type is determined based on the threat level.

The HTN in Figure 18 plans a bounding over-watch movement if the answer is "ContactLikely," a traveling over-watch movement if the answer is "ContactPossible," and a traveling movement if the answer is "ContactNotLikely." This decision conforms to the Army Field Manual FM 3–21.10 Chapter 3.

### 2. Ontology

Two OWL ontologies were created using Protégé 4.3: Contact.owl and IED.owl. The steps described in Chapter II were followed. Figures 19 and 20 display snapshots of the different classes in the IED and Contact ontologies, respectively, in Protégé 4.3. Figure 21 shows a detailed explanation of the IED ontology. There are three subtypes of an IED based on the delivery method. It is a suicide bomber IED if the delivery method is a kind of a human. It is a vehicle borne IED if the delivery method is a kind of vehicle. It is a packaged IED if it is not one of the previous subtypes. The severity of each subtype is determined based on the number of indicators seen in the environment. A high impact IED has at least three indicators, a medium impact IED has two indicators, and a low

impact IED has exactly one indicator. The user can add any other type by defining a new subclass in the ontology. No other changes are needed in COMBATXII or in the connecting mechanism. This is one of the advantages to use of ontologies that will be discussed in more detail later; that is, analysts can adjust the decision-making more easily by modifying the data model (ontology) rather than modifying the software logic.

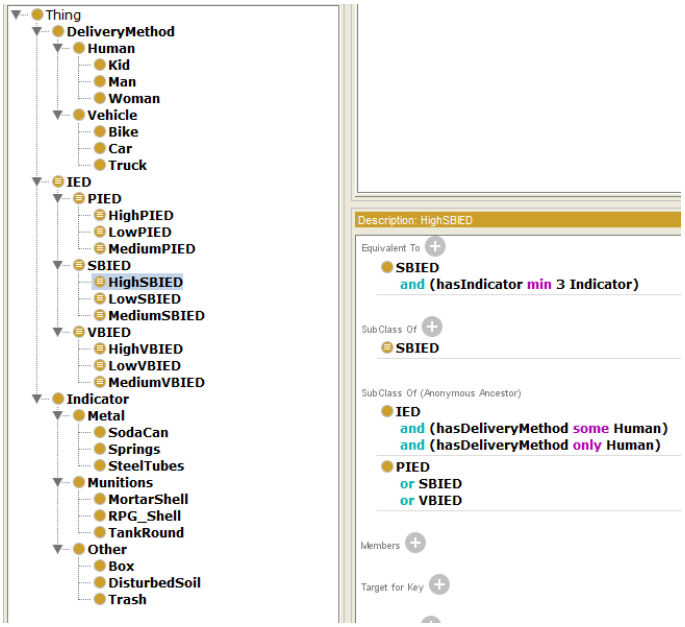


Figure 19. IED ontology in Protégé 4.3, showing the class definition for the HighSBIED class

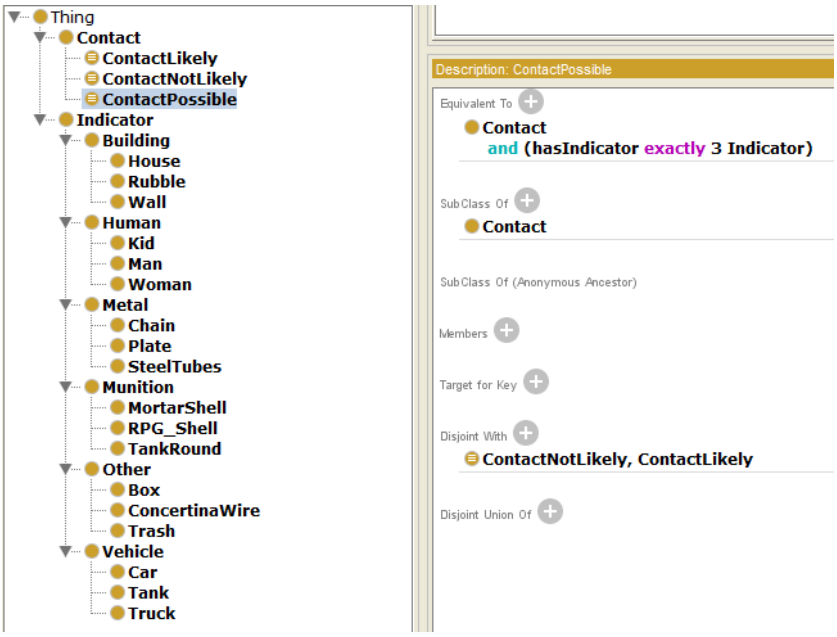


Figure 20. Contact ontology in Protégé 4.3, showing the class definition for the ContactPossible class

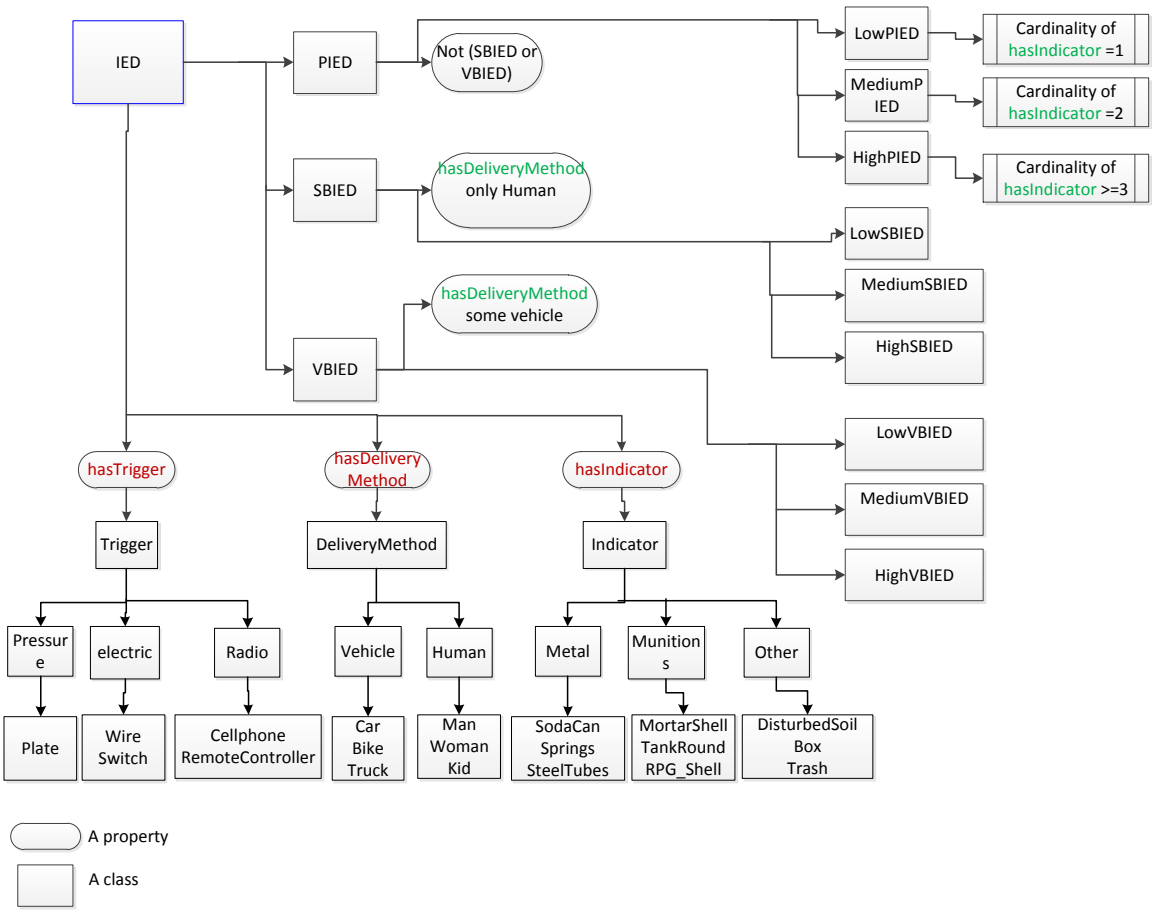


Figure 21. Structure of the IED ontology

## 3. Java Connection

Initially, the implementation used Protégé 3.5 for the ontology, the reasoner Pellet, and the Protégé-OWL API for the connecting mechanism, and everything worked fine. However, this did not scale well even with one property. When the number of classes and axioms were increased, the reasoner would sometimes fail and sometimes take an inordinate amount of time to perform the reasoning process, taking so long in other times. Table 3 shows the experiments conducted to study the scalability for the IED ontology. The aforementioned observations were not consistent, but generally from Test 4 the system becomes unreliable. The reasoner was able to classify the ontology but not compute the inferred types. Explaining the aforementioned behavior and investigating how to increase the performance of reasoners is beyond the scope of this thesis. However,

there are different ways to try to address this issue: change the reasoner or move to a different Protégé version

Table 3. Testing the scalability issue

| | |
|---|---|
| Test1: IED ontology type 1: | (3 IED classes and 3 indicator classes) |
| Test2: IED ontology type 2: | (3 IED classes and 6 indicator classes) |
| Test3: IED ontology type 3: | (4 IED classes and 3 indicator classes) |
| Test4: IED ontology type 4: | (4 IED classes and 6 indicator classes) |
| Test5: IED ontology type 5: | (5 IED classes and 3 indicator classes) |
| Test6: IED ontology type 6: | (5 IED classes and 6 indicator classes) |
| Test7: IED ontology type 7: | (6 IED classes and 3 indicator classes) |
| Test8: IED ontology type 8: | (6 IED classes and 6 indicator classes) |

Changing to Protégé 4.3, the OWL API, and the Hermit reasoner, the performance improved, and the reasoner did not fail for increased numbers of classes and properties. The Java code is completely different from the first implementation because the API changed.

The Java Implementation contains 3 classes:

TestMain: Gets an array of strings from the user and calls the main program. This emulates the message that would be received from COMBATXXI. Therefore, this class does not need to be included in the JAR file.

Main:Calls all other code.

Ontology: Initializes the ontology, gets a reasoner, and creates instances of the OWL API classes and interfaces:

1. Get the main class

2. Create an anonymous class as a subclass of the main class(this will be the answer to our question)

3. Based on the strings passed, find the links between them and the main class and add axioms to the anonymous class

Figure 22 better explains this approach.Any ontology can be represented as a graph of classes and properties linked to each other. Class0 is the main class. The program creates an anonymous subclass of Class0. The strings passed are Class1, Class2, Class3, and Class4. The program then finds property0, property1, and property2.Next, it adds these properties to the anonymous class. Finally, the reasoner determines the type of the anonymous class.Classes X and properties X are added to show that there are other classes and properties that do not link the anonymous class to the passed strings. For more details, refer to the code with inline comments in Appendix A.

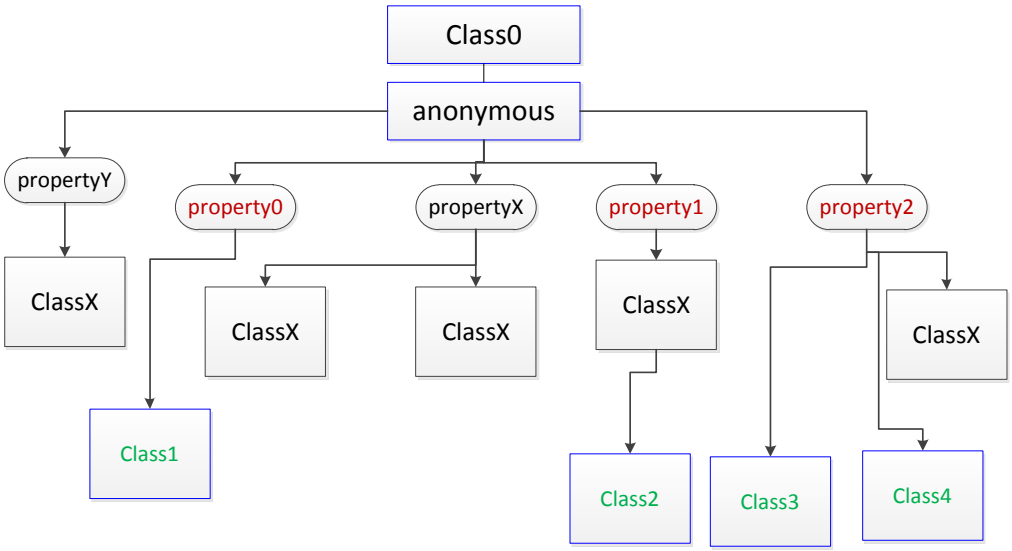


Figure 22. Anonymous Ontology

To replicate the connection, use a Java integrated development environment (IDE) such as Eclipse or Netbeans to create a new project. Then, add external jars (the plugins of Protégé 4.3 which are also the OWL API and the reasoner).Create the two classes Main and Ontology. Finally, create a JAR file consisting of these classes.

The code is generic and not specific to any ontology for the following reasons. First, based on the strings passed, it knows which ontology to load. Second, an unknown string is ignored. Third, if the ontology developer used the covering axiom feature provided by Protégé, the answer is guaranteed to return a type.

## B. RESULTS

Figures 23, 24, and 25 show the results from COMBATXXI output console of the IED type behavior, and Figures 26, 27, and 28 show the results from COMBATXXI output console of the level-of-threat behavior. The examples show that the reasoner provides a different answer when the input is changed.

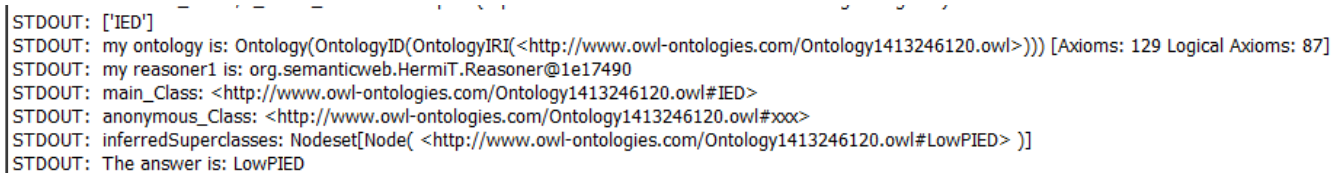


Figure 23. COMBATXXI output: determining the IED-type when the entity sees nothing. The answer is low probability packaged IED (LowPIED).

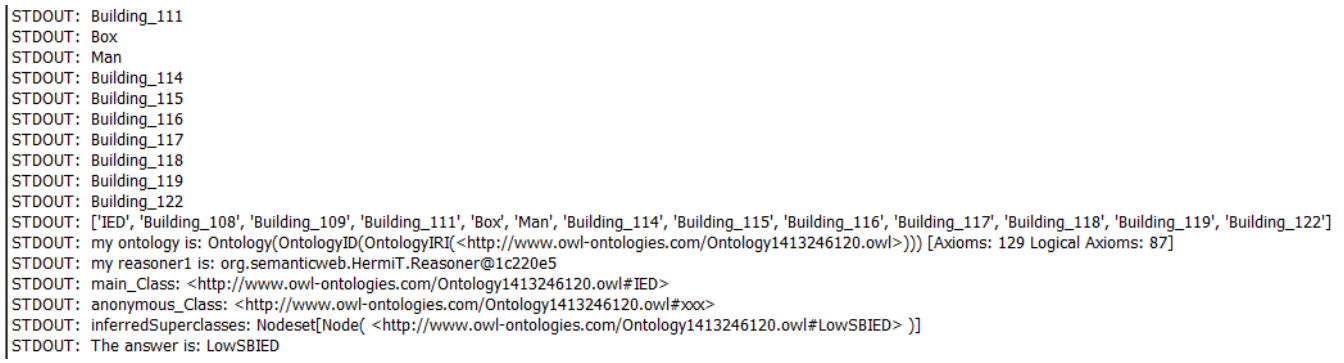


Figure 24. COMBATXXI output: determining the IED-type when the entity sees some buildings, a man, and a box. The answer is low probability suicide bomber IED (LowSBIED).

```
STDOUT:  Building_91
STDOUT:  Trash
STDOUT:  Building_133
STDOUT:  Truck
STDOUT:  MortarShell
STDOUT:  ['IED', 'Building_91', 'Trash', 'Building_133', 'Truck', 'MortarShell']
STDOUT:  my ontology is: Ontology(OntologyID(OntologyIRI(<http://www.owl-ontologies.com/Ontology1413246120.owl>))) [Axioms: 129 Logical Axioms: 87]
STDOUT:  my reasoner1 is: org.semanticweb.HermiT.Reasoner@13d953d
STDOUT:  main_Class: <http://www.owl-ontologies.com/Ontology1413246120.owl#IED>
STDOUT:  anonymous_Class: <http://www.owl-ontologies.com/Ontology1413246120.owl#xxx>
STDOUT:  inferredSuperclasses: Nodeset[Node( <http://www.owl-ontologies.com/Ontology1413246120.owl#MediumVBIED> )]
STDOUT:  The answer is: MediumVBIED
```

Figure 25. COMBATXXI output: determining the IED-type when the entity sees some buildings, trash, mortar shell and a truck. The answer is medium probability vehicle borne IED (MediumVBIED).

```
STDOUT:  ['Contact']
STDOUT:  my ontology is: Ontology(OntologyID(OntologyIRI(<http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact>))) [Axioms: 70 Logical Axioms: 40]
STDOUT:  my reasoner1 is: org.semanticweb.HermiT.Reasoner@19d3b6f
STDOUT:  main_Class: <http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Contact>
STDOUT:  anonymous_Class: <http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#xxx>
STDOUT:  inferredSuperclasses: Nodeset[Node( <http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#ContactNotLikely> )]
STDOUT:  The answer is: ContactNotLikely
```

Figure 26. COMBATXXI output: determining the level-of-threat when the entity sees nothing. The answer is ContactNotLikely.

```
STDOUT: Building_91
STDOUT: Trash
STDOUT: Building_133
STDOUT: Truck
STDOUT: MortarShell
STDOUT: ['Contact', 'Building_91', 'Trash', 'Building_133', 'Truck', 'MortarShell']
STDOUT: my ontology is: Ontology(OntologyID(OntologyIRI(<http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact>))) [Axioms: 70 Logical Axioms: 40]
STDOUT: my reasoner1 is: org.semanticweb.HermiT.Reasoner@209f2
STDOUT: main_Class: <http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Contact>
STDOUT: anonymous_Class: <http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#xxx>
STDOUT: inferredSuperclasses: Nodeset[Node( <http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#ContactPossible> )]
STDOUT: The answer is: ContactPossible
```

Figure 27. COMBATXXI output: determining the level-of-threat when the entity sees some buildings, trash, truck, and a mortar shell. The answer is ContactPossible.

```
STDOUT: Box
STDOUT: Man
STDOUT: Building_114
STDOUT: Building_115
STDOUT: Truck
STDOUT: Building_117
STDOUT: Plate
STDOUT: Building_119
STDOUT: Building_122
STDOUT: ['Contact', 'Building_108', 'Building_109', 'Building_111', 'Box', 'Man', 'Building_114', 'Building_115', 'Truck', 'Building_117', 'Plate', 'Building_119', 'Building_122']
STDOUT: my ontology is: Ontology(OntologyID(OntologyIRI(<http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact>))) [Axioms: 70 Logical Axioms: 40]
STDOUT: my reasoner1 is: org.semanticweb.HermiT.Reasoner@5ab21
STDOUT: main_Class: <http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Contact>
STDOUT: anonymous_Class: <http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#xxx>
STDOUT: inferredSuperclasses: Nodeset[Node( <http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#ContactLikely> )]
STDOUT: The answer is: ContactLikely
```

Figure 28. COMBATXXI output: determining the level-of-threat behavior when the entity sees some buildings, man, truck, and a metal plate. The answer is ContactLikely.

## C. ANALYSIS

The results proved the feasibility of using a knowledge representation system with inference capability in a closed form combat simulation. The IED and the threat type information were implemented in a knowledge representation system, and the connection to COMBATXXI was kept automatic to preserve the closed-form property of the model.

### 1. Advantages over Normal Programming

The new method is easier to maintain, because the specification is done in Protégé while the implementation is done in COMBATXXI. That is, the cognitive processing of entities can change based on user changes to the ontology, rather than on making changes to code. Moreover, the ontology and inference capabilities can be tested externally from the model.This is much more amenable to analyst activity than having to make code changes or do detailed testing of coded logic in the model.

Besides separation of specification and implementation, Protégé offers implicit definition of objects which creates loosely coupled components that are difficult to design in normal programming. For example, in Protégé we can define an object that is white, can be decomposed into more than five elements, and can move faster than 20 km/h without worrying about what these objects are. Then, after a simple object manipulation such as addition, removal, and modification, the reasoner can classify the new object in a few lines of code. Behind the scene, Protégé uses its reasoners to solve the problem for us. This is the advantage of using an external knowledge representation and reasoning over normal programming. This method gives us the capability of these well-established reasoners for free.

### 2. Sensing of Partial Information

After modeling a sensor, the results are sometimes not useful. With this new method, we can use partial information that was not useful by itself, but by combining it with other partial information, and infer some useful information. For example, when an entity observes some trash, it cannot make any useful conclusion from this observation. However, the reasoner can relate this information to another previous observation and

infer the likelihood of an IED. That is, any level of reasoning can be performed based on the declarative knowledge base and information available from the model.

### 3. Limitations

To fully benefit from the new method, the simulation system has to have enough objects to reason about, and enough sensors capable of accessing those objects, because we cannot reason on information that is not available. In addition, there has to be a process to add objects and sensors to meet the requirement of new reasoning capabilities, because even a system with a plenty of objects and sensors cannot include everything. For example, if an analyst wants to introduce new reasoning capabilities, they need to enumerate all objects they need for their specific scenario. Then, on the COMBATXXI side, the scenario builder needs to create those objects and allow entities to see them or create sensors to access them. The IED example in this thesis needed several objects and sensors that could not be constructed due to time constraint. Therefore, the author had to create notional objects in the model to mimic the real objects.

# V. CONCLUSION

The author proved the feasibility of using a knowledge representation system with inference capability in a closed form combat simulation. This new method provides many advantages over normal programming such as better separation of specification in the knowledge base and implementation in the simulation system, creation of implicit loosely-coupled object, and less work to accomplish the same thing since Protégé is using well-established reasoners for free.

However, the current thesis did not test that COMBATXXI entities have a dynamic reasoning as their knowledge of the battle space grows and changes, but it achieved the initial step toward this goal because the current system has the capability to grow the knowledge representation. Growing the knowledge space can be achieved by saving the ontology in the connecting mechanism by simply adding one line of code: OWLOntologyManager.saveOntology (ontology, format, IRI).That is, future work should save the ontology and query it more than once to test the dynamic reasoning as entities increase their knowledge of the battle space.

In addition, the author did not need to create individuals or data properties, because the implementation is only an example to show the feasibility of the connection between Protégé and COMBATXXI. More complex cases may need to create individuals and data properties. Currently, the connecting mechanism treats the arguments as classes, and only uses object properties. The addition of individuals and data properties can be added in a similar way. Currently, the strings received from COMBATXXI should be the classes in our ontologies. For example, the strings “IED,” “box,” “disturbed earth” are the names of classes in our ontology. A parser/mapper class can be added in future work to receive any string from COMBATXXI and make the necessary changes to that string in order to be understood by the ontology.

This thesis covered the types of questions: *What*? *What kind*? *How*? That is, an entity can reason over environment objects using these questions to get a single string back. For example, if an entity wants to know *what* it observed, the answer can be

friendly, neutral, or enemy. Second, if the entity knows there is something and wants to know its *kind* such as the kind of an IED, the response canbe "lowVBIED," "mediumPIED," or "highSVBIED." Third, the entity wants to know *how* to move. The response can be line, wedge, or bounding over-watch. All these questions can be answered without modification of the connecting mechanism, but obviously there are other useful questions that need to be addressed. Future work could investigate the following questions: Yes/No questions? What are the elements of something? How many? That is, an entity asking these questions receives a Boolean value, a list, or a number. For example, did friendly forces clear the building? Is the largest enemy unit within range? What are the cover and conceal positions in this terrain? How many terrorists are expected in that building? As this thesis did, these questions can be classified into groups. Each group should be implemented in the connecting mechanism.

Finally, follow-on research can work towards more cognitive modeling in order to distinguish between manned systems and unmanned systems in simulations, since as of now, every system in simulations has only robotic behavior.

# APPENDIX A. CONNECTING MECHANISM

```
package owlAPI;

import java.util.ArrayList;
import java.util.Collection;
import java.util.Set;
import org.semanticweb.owlapi.model.OWLClass;
import org.semanticweb.owlapi.reasoner.*;

public class Main {

   //just organizes the show by calling other methods, and returns the answer

   public Collection<String> getAnswer(String[] arr) throws Exception {
      //determine and initialize the ontology and a reasoner
      Ontology ontology = new Ontology();
      ontology.init(arr);

      //construct the anonymous class
      OWLClass IED_Class = ontology.prepareClass(arr);
      //System.out.println("IED_Class: " + IED_Class);

      //reasoner2 was needed because reasoner1 was used inside ontology...
      //if used Again, it does reason over gettingsupperClasses and the answer is thing
      OWLReasoner reasoner = ontology.getReasoner();
      //System.out.println("my reasoner is: " + reasoner);

      //ask reasoner2 to classify the anonymous class
      NodeSet<OWLClass> inferredSuperclasses = reasoner.getSuperClasses(IED_Class,
true);
      System.out.println("inferredSuperclasses: " + inferredSuperclasses);

      //put the superclasses in a collection (arraylist) instead of nodeSet
      Collection<String> myStr = new ArrayList<String>();
      for (Node<OWLClass> node : inferredSuperclasses) {
         Set<OWLClass> entities = node.getEntities();
         for (OWLClass entity : entities) {
            myStr.add(entity.getIRI().getFragment());
            //System.out.println("The answer is: " + myStr);

         }
      }
      return myStr;
```

```
    }
}



package owlAPI;

import java.io.File;
import java.net.URL;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Set;
import org.semanticweb.HermiT.Reasoner;
import org.semanticweb.owlapi.apibinding.OWLManager;
import org.semanticweb.owlapi.model.*;
import org.semanticweb.owlapi.reasoner.*;

public class Ontology {

    private OWLDataFactory factory;
    private OWLOntology ontology;
    private OWLOntologyManager manager;
    private Collection<OWLObjectProperty> collectionProps = new
ArrayList<OWLObjectProperty>();
    private OWLReasoner reasoner;

   //#########################################################################
//#                                                                       #
//#########################################################################
    public void init(String[] arr) throws OWLOntologyCreationException {

        //in combatxxi, the first element should be always the name of the ontology
        String ontologyName = arr[0];
        //specify the absolute or the relative path for owl files
        //File file = new File("E:/thesis/Ontology/Testing/"+ontologyName+."owl");
        URL url = getClass().getResource("/owlAPI/files/" + ontologyName + ."owl");
        File file = new File(url.getPath());

        //create OWLManager. it will be used to load ontology and create
        //OWLFactory
        manager = OWLManager.createOWLOntologyManager();
        //load ontolgy
        ontology = manager.loadOntologyFromOntologyDocument(file);
                //System.out.println("my ontology is: " + ontology);
```

```
        //create owlFactory
        factory = manager.getOWLDataFactory();
        //create reasonerFactory
        OWLReasonerFactory reasonerFactory = new Reasoner.ReasonerFactory();
        //create reasoner
        reasoner = reasonerFactory.createReasoner(ontology);
        System.out.println("my reasoner1 is: " + reasoner);
    }


//#########################################################################
 //#                                    prepare class                                    #
//#########################################################################
    public OWLClass prepareClass(String[] arr) {
        OWLOntologyID ontID = ontology.getOntologyID();
        IRI iri = ontID.getOntologyIRI();
        //get class from its iri (not directly from its ontology).
        //class iri is ontology iri+#className
        //here for simplicity I have main class name equals ontology name
        OWLClass main_Class = factory.getOWLClass(IRI.create(iri + "#" + arr[0]));
        System.out.println("main_Class: " + main_Class);
        //this would create a class named xxx since it does not exist
        // xxx= the name does not matter because I am not saving the class
        //future work can save the classes to update the knowledge base
        OWLClass anonymous_Class = factory.getOWLClass(IRI.create(iri + "#xxx"));
        //make the anonymous class a subclass of the main class
        //we start by creating an axiom
        OWLAxiom axiom = factory.getOWLSubClassOfAxiom(anonymous_Class,
main_Class);
        //then we add the axiom to the ontology
        AddAxiom addAxiom = new AddAxiom(ontology, axiom);
        // We now use the manager to apply the change
        manager.applyChange(addAxiom);
              //to save it permanently save the ontology at the end.

        //get all object properties (in the entire ontology)
        //only some properties pertain to our anonymous class
        Set<OWLObjectProperty> props = ontology.getObjectPropertiesInSignature();
                    //System.out.println("props"+props);

        //use private method to add object properties that have main class in domain
        //to collectionProps collection.
        getProps(props, main_Class);
        //System.out.println("collectionProps"+collectionProps);
```

```
OWLClass[] namedClass = new OWLClass[arr.length - 1];
//*************************************************************
//*      add related subclasses to anonymous class          *
//*************************************************************
//loop through these related properties
for (OWLObjectProperty prop : collectionProps) {
   int propCounter = 0;
   // System.out.println("the property: "+prop);
   //for earch property: get ranges
   Set<OWLClassExpression> ranges = prop.getRanges(ontology);
   //System.out.println(range);
   //for each given class (passed from Combatxxi) see if it is related to
   //this property. i.e it is in the range or a subclass of something in the range
   for (int i = 0; i < arr.length - 1; i++) {
      //i+1 because we already got arr[0] which is namedClass0.
      namedClass[i] = factory.getOWLClass(IRI.create(iri + "#" + arr[i + 1]));
      //System.out.println("----The investigated namedClass[i]: "+namedClass[i]);

      //for each class: find if it is a subclass (descendant) of the range of this property
      for (OWLClassExpression range : ranges) {

         //----------for each range get the subclasses
         NodeSet<OWLClass> subclasses = reasoner.getSubClasses(range, false);
         //System.out.println("subclasses of range: "+subclasses);
         //for each node in subclasses
         for (Node<OWLClass> subclassNode : subclasses) {
            //get subclasses from the nodes
            Set<OWLClass> entities = subclassNode.getEntities();
            //compare each subclass to given class from cxxi
            for (OWLClassExpression subclass : entities) {
               ////System.out.println("subclass: "+subclass);
               if (subclass != null && subclass instanceof OWLClass) {
                  if (((OWLClass) subclass).getIRI().equals(namedClass[i].getIRI())) {
                     // if yes, increment a counter for this property and update
                     //anonymousClass with "some" restriction for this property

                     propCounter++;
                     //System.out.println("--------propCounterInside: "+propCounter);
                     //create hasSomeValue expression
                     OWLClassExpression haspropClass =
factory.getOWLObjectSomeValuesFrom(prop, namedClass[i]);

                     // make the anonymous class a subclass of a class that has
```

```
                        //the previous expression through an axiom
                        OWLSubClassOfAxiomax =
factory.getOWLSubClassOfAxiom(anonymous_Class, haspropClass);

                        // Add the axiom to our ontology
                        AddAxiom addAx = new AddAxiom(ontology, ax);
                        manager.applyChange(addAx);
                    }
                }
              }//end for suclass in entities
            }//end node class in nodeset of classes
          }//end ranges
        }//end namedClass in the array

        //update anonymousClass with "exactly" restriction for this property and its range
(using counter)
        //notice that if a property did not show up, its propCounter will 0 and it
        //won't be added to cardinality expression.
        OWLObjectExactCardinality hasExactpropClass =
factory.getOWLObjectExactCardinality(propCounter, prop);
        // make the anonymous class a subclass of a class that has
        //the previous expression through an axiom
        OWLSubClassOfAxiom ax2 =
factory.getOWLSubClassOfAxiom(anonymous_Class, hasExactpropClass);
        // Add the axiom to our ontology
        AddAxiom addAx2 = new AddAxiom(ontology, ax2);
        manager.applyChange(addAx2);

      }//end property in propCollection

      return anonymous_Class;
    }

    //input: set of properties and a class
    //output: collection of properties related to given class
    private void getProps(Set<OWLObjectProperty> props, OWLClass main_Class) {
      //for each property
      for (OWLObjectProperty prop : props) {
        //exclude properties without domains
        if (!prop.getDomains(ontology).isEmpty()) {
          //now for each domain of property
          for (OWLClassExpression od : prop.getDomains(ontology)) {
            if (od != null && od instanceof OWLClass) {
              //if one of these domains equals the main class we want
              //this property
```

```
                    if (((OWLClass) od).getIRI().equals(main_Class.getIRI())) {
                        collectionProps.add(prop);
                    }
                }
            }
        }
    }
}


//#############################################################################
  //#                                    getReasoner                          #
 //#############################################################################
    public OWLReasoner getReasoner() {
        OWLReasonerFactory reasonerFactory = new Reasoner.ReasonerFactory();
        OWLReasoner reasoner2 = reasonerFactory.createReasoner(ontology);
        //System.out.println("my reasoner2 is: " + reasoner2);
        return reasoner2;
    }

}
```

# APPENDIX B. OWL FILES

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
<!ENTITY owl "http://www.w3.org/2002/07/owl#" >
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>


<rdf:RDF
xmlns="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#"
    xml:base="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<owl:Ontology
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact"/>



<!--
   ///////////////////////////////////////////////////////////////////////////////////
   //
   // Object Properties
   //
   ///////////////////////////////////////////////////////////////////////////////////
    -->




<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#hasIndicator --
>

<owl:ObjectProperty
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#hasIndi
cator">
```

<rdfs:domain
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Contact"/>
<rdfs:range
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Indicator"/>
</owl:ObjectProperty>

<!--
///////////////////////////////////////////////////////////////////////////////////
//
// Classes
//
///////////////////////////////////////////////////////////////////////////////////
-->

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Box -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Box">
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Other"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Building -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Building">
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Indicator"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Car -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Car">
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Vehi
cle"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Chain -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Chain"
>
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Met
al"/>
</owl:Class>

<!--
http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#ConcertinaWire --
>

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Concer
tinaWire">
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Othe
r"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Contact -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Contac
t"/>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#ContactLikely
-->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Contac
tLikely">
<owl:equivalentClass>
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Contac
t"/>
<owl:Class>
<owl:complementOf>
<owl:Class>
<owl:unionOf rdf:parseType="Collection">
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Contac
tNotLikely"/>
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Contac
tPossible"/>
</owl:unionOf>
</owl:Class>
</owl:complementOf>
</owl:Class>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Con
tact"/>
</owl:Class>

<!--
http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#ContactNotLikely
-->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Contac
tNotLikely">
<owl:equivalentClass>
<owl:Class>

<owl:intersectionOf rdf:parseType="Collection">
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Contact"/>
<owl:Restriction>
<owl:onProperty
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#hasIndicator"/>
<owl:onClass
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Indicator"/>
<owl:maxQualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">2</owl:maxQualifiedCardinality>
</owl:Restriction>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Contact"/>
</owl:Class>

<!--
http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#ContactPossible
-->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#ContactPossible">
<owl:equivalentClass>
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Contact"/>
<owl:Restriction>
<owl:onProperty
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#hasIndicator"/>
<owl:onClass
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Indicator"/>

<owl:qualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">3</owl:qualifiedCardinality>
</owl:Restriction>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Contact"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#House -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#House">
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Building"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Human -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Human">
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Indicator"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Indicator -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Indicator"/>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Kid -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Kid">
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Hu
man"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Man -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Man">
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Hu
man"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Metal -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Metal"
>
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Indi
cator"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#MortarShell --
>

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Mortar
Shell">
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Mun
ition"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Munition -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Munition">
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Indicator"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Other -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Other">
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Indicator"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Plate -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Plate">
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Metal"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#RPG_Shell -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#RPG_Shell">

<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Munition"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Rubble -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Rubble">
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Building"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#SteelTubes -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#SteelTubes">
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Metal"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Tank -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Tank">
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Vehicle"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#TankRound -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#TankR
ound">
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Mun
ition"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Trash -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Trash"
>
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Othe
r"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Truck -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Truck"
>
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Vehi
cle"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Vehicle -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Vehicl
e">
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Indi
cator"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Wall -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Wall">
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Building"/>
</owl:Class>

<!-- http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Woman -->

<owl:Class
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Woman">
<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Human"/>
</owl:Class>

<!--
///////////////////////////////////////////////////////////////////////////////////
//
// General axioms
//
///////////////////////////////////////////////////////////////////////////////////
-->

<rdf:Description>
<rdf:type rdf:resource="&owl;AllDisjointClasses"/>
<owl:members rdf:parseType="Collection">
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#MortarShell"/>
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#RPG_Shell"/>
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#TankRound"/>
</owl:members>

</rdf:Description>
<rdf:Description>
<rdf:type rdf:resource="&owl;AllDisjointClasses"/>
<owl:members rdf:parseType="Collection">
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Box"/>
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Concer
tinaWire"/>
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Trash"/
>
</owl:members>
</rdf:Description>
<rdf:Description>
<rdf:type rdf:resource="&owl;AllDisjointClasses"/>
<owl:members rdf:parseType="Collection">
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#House"
/>
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Rubble
"/>
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Wall"/
>
</owl:members>
</rdf:Description>
<rdf:Description>
<rdf:type rdf:resource="&owl;AllDisjointClasses"/>
<owl:members rdf:parseType="Collection">
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Chain"
/>
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Plate"/
>
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#SteelT
ubes"/>
</owl:members>
</rdf:Description>
<rdf:Description>
<rdf:type rdf:resource="&owl;AllDisjointClasses"/>
<owl:members rdf:parseType="Collection">

<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Human
"/>
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Metal"/
>
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Muniti
on"/>
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Other"/
>
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Vehicl
e"/>
</owl:members>
</rdf:Description>
<rdf:Description>
<rdf:type rdf:resource="&owl;AllDisjointClasses"/>
<owl:members rdf:parseType="Collection">
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Car"/>
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Tank"/
>
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Truck"
/>
</owl:members>
</rdf:Description>
<rdf:Description>
<rdf:type rdf:resource="&owl;AllDisjointClasses"/>
<owl:members rdf:parseType="Collection">
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Kid"/>
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Man"/
>
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Woma
n"/>
</owl:members>
</rdf:Description>
<rdf:Description>
<rdf:type rdf:resource="&owl;AllDisjointClasses"/>

<owl:members rdf:parseType="Collection">
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Contac
tLikely"/>
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Contac
tNotLikely"/>
<rdf:Description
rdf:about="http://www.semanticweb.org/localadmin/ontologies/2014/12/Contact#Contac
tPossible"/>
</owl:members>
</rdf:Description>
</rdf:RDF>

<!-- Generated by the OWL API (version 3.4.2) http://owlapi.sourceforge.net -->

<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
<!ENTITY owl "http://www.w3.org/2002/07/owl#" >
<!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
<!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#" >
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
<!ENTITY protege "http://protege.stanford.edu/plugins/owl/protege#" >
<!ENTITY xsp "http://www.owl-ontologies.com/2005/08/07/xsp.owl#" >
]>

<rdf:RDF xmlns="http://www.owl-ontologies.com/Ontology1413246120.owl#"
    xml:base="http://www.owl-ontologies.com/Ontology1413246120.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<owl:Ontology rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl"/>

<!--
///////////////////////////////////////////////////////////////////////////////
//
// Object Properties
//
///////////////////////////////////////////////////////////////////////////////
 -->

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#hasDeliveryMethod -->

<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#hasDeliveryMethod">
<rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#DeliveryMethod"/>
<rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#IED"/>
</owl:ObjectProperty>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#hasIndicator -->

<owl:ObjectProperty rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#hasIndicator">
<rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#IED"/>
<rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator"/>
</owl:ObjectProperty>

<!--
///////////////////////////////////////////////////////////////////////////////
//
// Classes
//
///////////////////////////////////////////////////////////////////////////////
 -->

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#Bike -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#Bike">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Vehicle"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#Box -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#Box">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Other"/>
<owl:disjointWith rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#DisturbedSoil"/>
<owl:disjointWith rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Trash"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#Car -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#Car">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Vehicle"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#DeliveryMethod -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#DeliveryMethod">
<owl:disjointWith rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#IED"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#DisturbedSoil -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#DisturbedSoil">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Other"/>
<owl:disjointWith rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Trash"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#HighPIED -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#HighPIED">
<owl:equivalentClass>
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#PIED"/>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasIndicator"/>
<owl:onClass rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator"/>
<owl:minQualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">3</owl:minQualifiedCardinality>
</owl:Restriction>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#PIED"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#HighSBIED -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#HighSBIED">
<owl:equivalentClass>

<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#SBIED"/>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasIndicator"/>
<owl:onClass rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator"/>
<owl:minQualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">3</owl:minQualifiedCardinality>
</owl:Restriction>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#SBIED"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#HighVBIED -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#HighVBIED">
<owl:equivalentClass>
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#VBIED"/>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasIndicator"/>
<owl:onClass rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator"/>
<owl:minQualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">3</owl:minQualifiedCardinality>
</owl:Restriction>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#VBIED"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#Human -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#Human">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#DeliveryMethod"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#IED -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#IED">
<owl:equivalentClass>
<owl:Class>
<owl:unionOf rdf:parseType="Collection">
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#PIED"/>
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#SBIED"/>
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#VBIED"/>
</owl:unionOf>
</owl:Class>
</owl:equivalentClass>
<owl:disjointWith rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#IED1 -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#IED1">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#IED"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasIndicator"/>

<owl:onClass rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator"/>
<owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasIndicator"/>
<owl:someValuesFrom rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#SteelTubes"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasDeliveryMethod"/>
<owl:onClass rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#DeliveryMethod"/>
<owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasDeliveryMethod"/>
<owl:someValuesFrom rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Car"/>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#IED2 -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#IED2">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#IED"/>
<rdfs:subClassOf>
<owl:Restriction>

<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasIndicator"/>
<owl:onClass rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator"/>
<owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">0</owl:qualifiedCardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasDeliveryMethod"/>
<owl:someValuesFrom rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Man"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasDeliveryMethod"/>
<owl:onClass rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#DeliveryMethod"/>
<owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#IED3 -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#IED3">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#IED"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasIndicator"/>
<owl:onClass rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator"/>
<owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">0</owl:qualifiedCardinality>
</owl:Restriction>

</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasDeliveryMethod"/>
<owl:someValuesFrom rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Man"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasDeliveryMethod"/>
<owl:onClass rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#DeliveryMethod"/>
<owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">2</owl:qualifiedCardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasDeliveryMethod"/>
<owl:someValuesFrom rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Car"/>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#IED4 -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#IED4">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#IED"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasDeliveryMethod"/>
<owl:onClass rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#DeliveryMethod"/>
<owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">0</owl:qualifiedCardinality>

</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasIndicator"/>
<owl:onClass rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator"/>
<owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasIndicator"/>
<owl:someValuesFrom rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#DisturbedSoil"/>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#IED5 -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#IED5">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#IED"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasIndicator"/>
<owl:someValuesFrom rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Box"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasIndicator"/>
<owl:someValuesFrom rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Trash"/>
</owl:Restriction>

</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasDeliveryMethod"/>
<owl:onClass rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#DeliveryMethod"/>
<owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasDeliveryMethod"/>
<owl:someValuesFrom rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Human"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasIndicator"/>
<owl:onClass rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator"/>
<owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">2</owl:qualifiedCardinality>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator"/>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#Kid -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#Kid">

<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Human"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#LowPIED -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#LowPIED">
<owl:equivalentClass>
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#PIED"/>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasIndicator"/>
<owl:onClass rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator"/>
<owl:maxQualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxQualifiedCardinality>
</owl:Restriction>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#PIED"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#LowSBIED -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#LowSBIED">
<owl:equivalentClass>
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#SBIED"/>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasIndicator"/>

<owl:onClass rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator"/>
<owl:maxQualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxQualifiedCardinality>
</owl:Restriction>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#SBIED"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#LowVBIED -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#LowVBIED">
<owl:equivalentClass>
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#VBIED"/>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasIndicator"/>
<owl:onClass rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator"/>
<owl:maxQualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxQualifiedCardinality>
</owl:Restriction>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#VBIED"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#Man -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#Man">

<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Human"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#MediumPIED -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#MediumPIED">
<owl:equivalentClass>
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#PIED"/>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasIndicator"/>
<owl:onClass rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator"/>
<owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">2</owl:qualifiedCardinality>
</owl:Restriction>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#PIED"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#MediumSBIED -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#MediumSBIED">
<owl:equivalentClass>
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#SBIED"/>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasIndicator"/>

<owl:onClass rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator"/>
<owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">2</owl:qualifiedCardinality>
</owl:Restriction>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#SBIED"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#MediumVBIED -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#MediumVBIED">
<owl:equivalentClass>
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#VBIED"/>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasIndicator"/>
<owl:onClass rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator"/>
<owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">2</owl:qualifiedCardinality>
</owl:Restriction>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#VBIED"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#Metal -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#Metal">

<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator"/>
<owl:disjointWith rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Munitions"/>
<owl:disjointWith rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Other"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#MortarShell -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#MortarShell">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Munitions"/>
<owl:disjointWith rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#RPG_Shell"/>
<owl:disjointWith rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#TankRound"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#Munitions -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#Munitions">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator"/>
<owl:disjointWith rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Other"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#Other -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#Other">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Indicator"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#PIED -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#PIED">
<owl:equivalentClass>
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#IED"/>
<owl:Class>
<owl:complementOf>
<owl:Class>
<owl:unionOf rdf:parseType="Collection">
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#SBIED"/>
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#VBIED"/>
</owl:unionOf>
</owl:Class>
</owl:complementOf>
</owl:Class>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#IED"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#RPG_Shell -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#RPG_Shell">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Munitions"/>
<owl:disjointWith rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#TankRound"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#SBIED -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#SBIED">
<owl:equivalentClass>
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#IED"/>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasDeliveryMethod"/>
<owl:someValuesFrom rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Human"/>
</owl:Restriction>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasDeliveryMethod"/>
<owl:allValuesFrom rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Human"/>
</owl:Restriction>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#SodaCan -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#SodaCan">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Metal"/>
<owl:disjointWith rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Springs"/>
<owl:disjointWith rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#SteelTubes"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#Springs -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#Springs">

<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Metal"/>
<owl:disjointWith rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#SteelTubes"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#SteelTubes -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#SteelTubes">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Metal"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#TankRound -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#TankRound">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Munitions"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#Trash -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#Trash">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Other"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#Truck -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#Truck">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Vehicle"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#VBIED -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#VBIED">
<owl:equivalentClass>
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#IED"/>
<owl:Restriction>
<owl:onProperty rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#hasDeliveryMethod"/>
<owl:someValuesFrom rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Vehicle"/>
</owl:Restriction>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#Vehicle -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#Vehicle">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#DeliveryMethod"/>
</owl:Class>

<!-- http://www.owl-ontologies.com/Ontology1413246120.owl#Woman -->

<owl:Class rdf:about="http://www.owl-ontologies.com/Ontology1413246120.owl#Woman">
<rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1413246120.owl#Human"/>
</owl:Class>

```
<!--
   ///////////////////////////////////////////////////////////////////////////////
   //
   // General axioms
   //
   ///////////////////////////////////////////////////////////////////////////////
    -->

<rdf:Description>
<rdf:type rdf:resource="&owl;AllDisjointClasses"/>
<owl:members rdf:parseType="Collection">
<rdf:Description rdf:about="http://www.owl-
ontologies.com/Ontology1413246120.owl#HighVBIED"/>
<rdf:Description rdf:about="http://www.owl-
ontologies.com/Ontology1413246120.owl#LowVBIED"/>
<rdf:Description rdf:about="http://www.owl-
ontologies.com/Ontology1413246120.owl#MediumVBIED"/>
</owl:members>
</rdf:Description>
<rdf:Description>
<rdf:type rdf:resource="&owl;AllDisjointClasses"/>
<owl:members rdf:parseType="Collection">
<rdf:Description rdf:about="http://www.owl-
ontologies.com/Ontology1413246120.owl#PIED"/>
<rdf:Description rdf:about="http://www.owl-
ontologies.com/Ontology1413246120.owl#SBIED"/>
<rdf:Description rdf:about="http://www.owl-
ontologies.com/Ontology1413246120.owl#VBIED"/>
</owl:members>
</rdf:Description>
</rdf:RDF>



<!-- Generated by the OWL API (version 3.4.2) http://owlapi.sourceforge.net -->
```

# APPENDIX C. XML REPRESENTATION OF THE HIERARCHICAL TASK NETWORKS

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<HTNNode AllowMsg="true" Name="SquadMove" Type="DEFAULT">
        <Parent>null</Parent>
        <DataMap>
                <DataKey>start_loc,cxxi.model.objects.features.CMWayPoint,Way point at the start of the
road</DataKey>
                <DataKey>end_loc,cxxi.model.objects.features.CMWayPoint,Way point at the end of the
road</DataKey>
        </DataMap>
        <Code IsFile="false"/>
        <Import>from HTN import UtilityFuncsExp</Import>
        <HTNNode AllowMsg="true" Name="initInfo" Type="DEFAULT">
                <Parent>SquadMove</Parent>
                <Code IsFile="false">if _gt_activeNode.getVar("isInited") == None:
        _gt_activeNode.putVar("isInited," 1)
        _htn_precon_ret=1
</Code>
                <Import/>
                <HTNNode AllowMsg="true" Name="addReplanTriggers" Type="DEFAULT">
                        <Parent>initInfo</Parent>
                        <Code IsFile="false"># goal tracker events
goalContainer.getCurrentExecutingStack().addReplanTrigger("GoalTracker_FinishedMove")
</Code>
                        <Import/>
                </HTNNode>
                <HTNNode AllowMsg="true" Name="isCommander" Type="DEFAULT">
                        <Parent>initInfo</Parent>
                        <Code IsFile="false">if state.isCommander():
        _htn_precon_ret=1
</Code>
                        <Import/>
                        <HTNNode AllowMsg="true" Name="findIED_Type" Type="DEFAULT">
                                <Parent>isCommander</Parent>
                                <Code IsFile="false">start = _gt_activeNode.getParam("start_loc").getLocation()
end = _gt_activeNode.getParam("end_loc").getLocation()
printMessage(" ," True)
bldgs = UtilityFuncsExp.getBuildingsAlong(start, end, 15)
print "number of buildings ,"len(bldgs)

printMessage("after import," True)
arr=["IED"]
for j in bldgs:
        print j.getAssignedName()
        arr.append(str(j.getAssignedName()))
#m=test_Ontology()
#print arr
UtilityFuncsExp.answer(arr)

if len(bldgs) &gt; 3:
        _htn_precon_ret=1
</Code>
                                <Import/>
                        </HTNNode>
                        <HTNNode AllowMsg="true" Name="determineNextBehavior" Type="INTERRUPT">
                                <Parent>isCommander</Parent>
                                <Code IsFile="false">printMessage("MOVE IN FORMATION," True)

formationName = "INF_WEDGE"
startCM = _gt_activeNode.getParam("start_loc")
endCM = _gt_activeNode.getParam("end_loc")

# set the path to the goal
```

```
goalPath = “HTN/Trees/MoveInFormation.xml”

# add the goal to a unit
UtilityFuncsExp.addGoal(
   info.getMyAssignedName(),
   1.0,
   goalPath,
   [formationName, startCM, endCM],
   None)
</Code>
                                        <Import/>
                                        <HTNNode AllowMsg=“true” Name=“nextBehavior” Type=“INTERRUPT”>
                                                <Parent>determineNextBehavior</Parent>
                                                <Code IsFile=“false”>_htn_precon_ret=1</Code>
                                                <Import/>
                                        </HTNNode>
                                </HTNNode>
                        </HTNNode>
                        <HTNNode AllowMsg=“false” Name=“endInit” Type=“INTERRUPT”>
                                <Parent>initInfo</Parent>
                                <Code IsFile=“false”>printMessage(“INIT FINISHED,” True)
</Code>
                                <Import/>
                        </HTNNode>
                </HTNNode>
                <HTNNode AllowMsg=“true” Name=“events” Type=“DEFAULT”>
                        <Parent>SquadMove</Parent>
                        <Code IsFile=“false”/>
                        <Import/>
                        <HTNNode AllowMsg=“true” Name=“isGoalTrackerEvent” Type=“DEFAULT”>
                                <Parent>events</Parent>
                                <Code IsFile=“false”>if state.getLastTrigger().startswith(“doGoalTracker_”):
                _htn_precon_ret=1
</Code>
                                <Import/>
                                <HTNNode AllowMsg=“true” Name=“isFinishedMove” Type=“DEFAULT”>
                                        <Parent>isGoalTrackerEvent</Parent>
                                        <Code IsFile=“false”>if state.getLastTrigger() == “doGoalTracker_FinishedMove”:
                _htn_precon_ret=1
</Code>
                                        <Import/>
                                        <HTNNode AllowMsg=“true” Name=“finishedMove” Type=“GOAL”>
                                                <Parent>isFinishedMove</Parent>
                                                <Code IsFile=“false”>printMessage(“MISSION COMPLETE,” True)
</Code>
                                                <Import/>
                                        </HTNNode>
                                </HTNNode>
                        </HTNNode>
                </HTNNode>
</HTNNode>



<?xml version=“1.0” encoding=“UTF-8” standalone=“no”?>
<HTNNode AllowMsg=“true” Name=“SquadMove” Type=“DEFAULT”>
                <Parent>null</Parent>
                <DataMap>
                        <DataKey>start_loc,cxxi.model.objects.features.CMWayPoint,Way point at the start of the
road</DataKey>
                        <DataKey>end_loc,cxxi.model.objects.features.CMWayPoint,Way point at the end of the
road</DataKey>
                </DataMap>
                <Code IsFile=“false”/>
                <Import>from HTN import UtilityFuncsExp</Import>
                <HTNNode AllowMsg=“true” Name=“initInfo” Type=“DEFAULT”>
                        <Parent>SquadMove</Parent>
                        <Code IsFile=“false”>if _gt_activeNode.getVar(“isInited”) == None:
```

```
        _gt_activeNode.putVar("isInited," 1)
        _htn_precon_ret=1
</Code>
                <Import/>
                <HTNNode AllowMsg="true" Name="addReplanTriggers" Type="DEFAULT">
                        <Parent>initInfo</Parent>
                        <Code IsFile="false"># goal tracker events
goalContainer.getCurrentExecutingStack().addReplanTrigger("GoalTracker_FinishedMove")
</Code>
                        <Import/>
                </HTNNode>
                <HTNNode AllowMsg="true" Name="isCommander" Type="DEFAULT">
                        <Parent>initInfo</Parent>
                        <Code IsFile="false">if state.isCommander():
        _htn_precon_ret=1
</Code>
                        <Import/>
                        <HTNNode AllowMsg="true" Name="isOverWatch" Type="DEFAULT">
                                <Parent>isCommander</Parent>
                                <Code IsFile="false">start = _gt_activeNode.getParam("start_loc").getLocation()
end = _gt_activeNode.getParam("end_loc").getLocation()
printMessage(" ," True)
bldgs = UtilityFuncsExp.getBuildingsAlong(start, end, 15)
print "number of buildings ,"len(bldgs)

arr=["Contact"]
for j in bldgs:
        print j.getAssignedName()
        arr.append(str(j.getAssignedName()))
#m=test_Ontology()
#print arr
ans=UtilityFuncsExp.answer(arr)

print ans

_gt_activeNode.putVar("ans," ans)


if ans in ["ContactPossible,""ContactLikely"]:
        _htn_precon_ret=1
</Code>
                                <Import/>
                                <HTNNode AllowMsg="true" Name="isBoundingOverWatch" Type="DEFAULT">
                                        <Parent>isOverWatch</Parent>
                                        <Code IsFile="false">if _gt_activeNode.getVar("ans") ==
"ContactLikely":
        _htn_precon_ret=1</Code>
                                        <Import/>
                                        <HTNNode AllowMsg="true" Name="boundingOverWatch"
Type="INTERRUPT">
                                                <Parent>isBoundingOverWatch</Parent>
                                                <Code IsFile="false">#only the commanders bfv2 from bfvs,
entity2from sqd5, and entity7 from sq4 (will be later left flank)
# they will add goals to their units to go to boundingOverwatch
printMessage("MOVE IN BOUNDING OVERWATCH," True)

formationName = "INF_WEDGE"
startCM = _gt_activeNode.getParam("start_loc")
endCM = _gt_activeNode.getParam("end_loc")


# set the path to the goal
goalPath = "HTN/Trees/MoveInBoundingOverwatch.xml"

# add the bounding overwatch goals to all units
UtilityFuncsExp.addGoalToUnit(
  state.getCurrentUnit().getName(),
  1.0,
```

```
  goalPath,
  [formationName, startCM, endCM],
  None)</Code>
                                                                <Import/>
                                                        </HTNNode>
                                                </HTNNode>
                                                <HTNNode AllowMsg="true" Name="travelingOverwatch" Type="INTERRUPT">
                                                        <Parent>isOverWatch</Parent>
                                                        <Code IsFile="false">#only the commanders bfv2 from bfvs,
entity2from sqd5, and entity7 from sq4 (will be later left flank)
# they will add goals to their units to go to boundingOverwatch
printMessage("MOVE IN TRAVELING OVERWATCH," True)

formationName = "INF_WEDGE"
startCM = _gt_activeNode.getParam("start_loc")
endCM = _gt_activeNode.getParam("end_loc")
'''

# set the path to the goal
goalPath = "HTN/Trees/MoveInBoundingOverwatch.xml"

# add the bounding overwatch goals to all units
UtilityFuncsExp.addGoalToUnit(
  state.getCurrentUnit().getName(),
  1.0,
  goalPath,
  [formationName, startCM, endCM],
  None)

'''</Code>
                                                        <Import/>
                                                </HTNNode>
                                        </HTNNode>
                                        <HTNNode AllowMsg="true" Name="traveling" Type="INTERRUPT">
                                                <Parent>isCommander</Parent>
                                                <Code IsFile="false">printMessage("MOVE IN TRAVELING," True)

formationName = "INF_WEDGE"
startCM = _gt_activeNode.getParam("start_loc")
endCM = _gt_activeNode.getParam("end_loc")

# set the path to the goal
goalPath = "HTN/Trees/MoveInFormation.xml"
'''
# add the goal to a unit
UtilityFuncsExp.addGoal(
   info.getMyAssignedName(),
   1.0,
   goalPath,
   [formationName, startCM, endCM],
   None)
'''</Code>
                                                <Import/>
                                        </HTNNode>
                                </HTNNode>
                                <HTNNode AllowMsg="false" Name="endInit" Type="INTERRUPT">
                                        <Parent>initInfo</Parent>
                                        <Code IsFile="false">printMessage("INIT FINISHED," True)
</Code>
                                        <Import/>
                                </HTNNode>
                        </HTNNode>
                        <HTNNode AllowMsg="true" Name="events" Type="DEFAULT">
                                <Parent>SquadMove</Parent>
                                <Code IsFile="false"/>
                                <Import/>
                                <HTNNode AllowMsg="true" Name="isGoalTrackerEvent" Type="DEFAULT">
                                        <Parent>events</Parent>
```

```
                                <Code IsFile="false">if state.getLastTrigger().startswith("doGoalTracker_"):
        _htn_precon_ret=1
</Code>
                                <Import/>
                                <HTNNode AllowMsg="true" Name="isFinishedMove" Type="DEFAULT">
                                        <Parent>isGoalTrackerEvent</Parent>
                                        <Code IsFile="false">if state.getLastTrigger() == "doGoalTracker_FinishedMove":
        _htn_precon_ret=1
</Code>
                                        <Import/>
                                        <HTNNode AllowMsg="true" Name="finishedMove" Type="GOAL">
                                                <Parent>isFinishedMove</Parent>
                                                <Code IsFile="false">printMessage("MISSION COMPLETE," True)
</Code>
                                                <Import/>
                                        </HTNNode>
                                </HTNNode>
                        </HTNNode>
                </HTNNode>
</HTNNode>
```

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF REFERENCES


Abburu, S. (2012). A Survey on Ontology Reasoners and Comparison. *International Journal of Computer Applications*, *57*(17): 33–39.

Balogh, I. et al. (2014). Using hierarchical task networks to create dynamic behaviors in combat models. Class material for MV4502, Naval Postgraduate School, fall 2014.

Balogh, I. et al.(2012, June 11–14). Use of hierarchical task networks to model complex operational tasks in COMBATXXI [PowerPoint presentation]. 80th Military Operations Research Society Symposium, Mission Area Analysis Branch, Quantico, VA.

Buss, A. H., & Stork, K. A. (1996, November). Discrete-event simulation on the World Wide Web using Java. In *Proceedings of the 28th conference on Winter Simulation* (pp. 780–785). Washington, DC: IEEE Computer Society.

Buss, A. H. (2014). Discrete-event simulation modeling. Class material for MV3302, Naval Postgraduate School, fall 2014.

Brachman, R., & Levesque, H. (2004). *Knowledge representation and reasoning*. San Francisco, CA: Elsevier.

Chappell, D. A., & Jewell, T. (2002). *Java web services*. Sebastopol, CA: O'Reilly Media.

Childers, C. (2006*). Applying semantic web concepts to support net-centric warfare using the tactical assessment markup language TAML*(master's thesis). Retrieved from Calhounhttp://calhoun.nps.edu/bitstream/handle/10945/2770/06Jun_Childers.pdf?sequence=1

Choosing between versions of desktop Protégé. (2013). Retrieved March 1, 2015, from http://protegewiki.stanford.edu/wiki/Protege4Migration.

Erol, K., Hendler, J., & Nau, D. S. (1994). HTN planning: Complexity and expressivity. In AAAI (Vol. 94, pp. 1123–1128).

Erol, K., Hendler, J. A., & Nau, D. S. (1995). *Semantics for hierarchical task-network planning* (Report No. ISR-TR-95-9). Maryland University College Park Inst For Systems Research.

Extensible Markup Language. (2014). Retrieved December 7, 2014, from http://www.w3.org/XML.

Fishman, G. S. (1978). *Principles of discrete event simulation*. New York, NY: John Wiley & Sons

Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, M., Eriksson, H., ... & Tu, S. W. (2003). The evolution of Protégé: an environment for knowledge-based systems development. *International Journal of Human-Computer Studies*, 58(1), 89–123.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 199–220.

Hjelm, J. (2001). *Creating the semantic Web with RDF: Professional developer's guide*. New York, NY: John Wiley and Sons

Horridge, M., Knublauch, H., Rector, A., Stevens, R., & Wroe, C. (2004). A practical guide to building OWL ontologies using the Protégé-OWL plugin and CO-ODE Tools Edition 1.0. University of Manchester.

Horridge, M., Jupp, S., Moulton, G., Rector, A., Stevens, R., & Wroe, C. (2009). A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools Edition1. 2. The University of Manchester.

Horrocks, I., Motik, B., & Wang, Z. (2012). *The HermiT OWL Reasoner* (paper). Oxford University, Oxford, United Kingdom.

Jennings, N. R. (2001). An agent-based approach for building complex software systems. *Communications of the ACM*, *44*(4), 35–41.

Kelton, W. D., & Law, A. M. (2000). *Simulation modeling and analysis*. Boston, MA: McGraw Hill.

Luger, G. F. (2005). *Artificial intelligence: Structures and strategies for complex problem solving*. Harlow, England: Pearson education.

Macal, C. M., & North, M. J. (2005). Tutorial on agent-based modeling and simulation. In Proceedings of the 37th conference on Winter simulation (pp. 2–15).Orlando, FL: Winter Simulation Conference.

Mitra,N et al., (2007). *SOAP Version 1.2 Part 0: Primer (Second Edition). W3C Recommendation*. Retrieved on March 27, 2015, from http://www.w3.org/TR/2007/REC-soap12-part0-20070427/

Mylopoulos, J., & Levesque, H. (1983). An overview of knowledge representation. In GWAI-83 (pp. 143–157). Springer Berlin Heidelberg.

Noy, N. F., & McGuinness, D. L. (2001). *Ontology Development 101: A Guide to Creating Your First Ontology*. Stanford, CA: Stanford University.

Obrst, L., & Davis, M. (2006). *Semantic wave.* Retrieved on March 27, 2015 fromhttp://semanticommunity.info/@api/deki/files/7298/MDavis02092006.pdf

Passin, Thomas. (2004). *Explorer's Guide to the Semantic Web*. Greenwich, CT: Manning Publishers.

Richmond, Paul W., Blais, Curtis L., Nagle, Joyce A., Goerger, Niki C., Gates, Buhrman Q., Burk, Robin K., Willis, John, and Keeter, Robert. (2007). Standards for the Mobility Common Operational Picture (M-COP): Elements of Ground Vehicle Maneuver. ERDC TR-07-4. U.S. Army Corps of Engineers Engineer Research and Development Center.

Simmons, R. F. (1973). Modeling English conversations. In *Proceedings of the June 4–8, 1973, National Computer Conference and Exposition* (pp. 451–451).

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y. (2007). Pellet: A practical owl-dl reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, *5*(2), 51–53.

Smith, R. (1985*). Knowledge-Based Systems: Concepts, Techniques, Examples* [PowerPoint presentation]. Retrieved May 02, 2014 from http://www.reidgsmith.com/Knowledge-Based_Systems_-_Concepts_Techniques_Examples_08-May-1985.pdf.

Teters,J. (2013). *Enhancing Entity Level Knowledge Representation and Environmental Sensing in COMBATXXI Using UAS Systems* (master's Thesis). Retrieved from Calhoun http://calhoun.nps.edu/bitstream/handle /10945/37732/13Sep_Teters_James.pdf?sequence=1

Web Ontology Language (OWL). (2013). *Web Ontology Language*. Retrieved March 09, 2015, from http://www.w3.org/2001/sw/wiki/OWL

THIS PAGE INTENTIONALLY LEFT BLANK

# INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California